Improved Static Analysis of Parameterised Boolean Equation Systems using Control Flow Reconstruction

Jeroen J. A. Keiren, Wieger Wesselink, and Tim A. C. Willemse

Eindhoven University of Technology, The Netherlands

Abstract. We present a method for fighting the state space explosion of parameterised Boolean equation systems (PBESs); these essentially are systems of mutually recursive Boolean fixed point equations, parameterised with data. PBESs can encode equivalence checking problems and model checking problems for symbolic, process algebraic specifications. Our method essentially consists of three phases: (1) the control flow in the PBES is reconstructed, detecting control flow parameters that were encoded in the description of the process, as well as control flow parameters that were introduced during the construction of the PBES; (2) we use a data flow analysis based on the control flow in the PBES to detect irrelevant data parameters and (3) we reset those data parameters of the equations in the PBES that were found to be irrelevant. Our reduction preserves the solution to the PBES, and never increases the size of the underlying Boolean equation system. The reduction is evaluated using a number of case studies.

1 Introduction

Model checking and equivalence checking of concurrent systems is often hampered by the infamous state space explosion problem. A lot of research is therefore directed at combatting this combinatorial problem. First generating a state space underlying a complex, concurrent system, and reducing it a posteriori is often inefficient, since generating the state space can be quite time consuming. A *static analysis* of a symbolic representation of a system may be able to reduce the state space size a priori [5, 4]. For instance, in [11] the control flow in a system was used to analyse its data flow, leading to significant reductions compared to other known static analysis techniques.

In [3], the authors suggest that the effectiveness of the static analysis techniques can be improved by taking the properties to be verified into account during the analysis. However, it is not immediately clear how existing methods can be extended to do so in general. We solve this problem by conducting the static analysis on *Parameterised Boolean Equation Systems* [8]. These are fixed point equation systems that can be used for solving a variety of verification problems, such as model checking of first order μ-calculus formulae over (possibly infinite) labelled transition systems and equivalence checking of various behavioural equivalences on labelled transition systems, see [10] and the references therein. The encodings of such verification problems are typically such that those parts of the system that do not influence validity of the property that is checked are automatically excluded from the PBES during the translation. PBESs are typically instantiated [10] to *Boolean equation systems* or *parity games*, for which

solving is decidable [7]. The instantiation, which is akin to a state space exploration, may not always terminate as solving for PBESs is, in general, undecidable.

Our contribution is a novel static analysis method for PBESs. This method consists of three separate phases. First, we compute a control flow graph for a given PBES. Second, we use this control flow graph to determine which data parameters are relevant in a control flow location. The final step consists of assigning some fixed, default value to those data parameters that are *not* relevant for a control flow location.

The notion of a control flow graph for PBESs is not self-evident: a PBES does not have an obvious graph structure. Instead, the control flow is typically encoded in the parameters of the equations, which may come from both the property and the specification. In addition, equations in PBESs can be mutually recursive. This means that parameters of one equation may affect parameters in another equation.

We propose a notion of a control flow parameter that allows for identifying a meaningful control flow graph of a PBES. Moreover, we provide efficient heuristics for identifying control flow parameters. Using these parameters, we define two different types of control flow graphs. The first —global— control flow graph considers all control flow parameters, and the values these can take on, simultaneously. As its size is potentially exponential in the number of control flow parameters, we define a second —local— type of control flow graph consisting of one graph per control flow parameter that does not suffer from this problem. Here, we draw inspiration from [11].

For both types of control flow graph we define a dedicated data flow analysis that conservatively marks data parameters that may influence the solution of the PBES. By resetting data parameters that are not marked relevant to a default value as soon as possible, the size of the underlying Boolean equation system is reduced. We prove that both versions of our data flow analysis, and the consequent resetting of irrelevant data parameters, are sound: they preserve the solution of the equation system, and, therefore the answer to the encoded verification problem.

We implemented our reduction in the context of the mCRL2 toolset [2] and applied it to a set of examples, showing that it outperforms previous static analysis techniques for PBESs [9] and that reductions of about 90% of the size of the underlying Boolean equation systems can be achieved.

Related work. We take inspiration from [4], which presents static analysis for state spaces in general, and [11] where live variable analysis is applied to reduce state spaces. In the latter, a reconstruction of the control flow is described for symbolic descriptions of processes without mutual recursion.

Liveness analysis techniques are well-known in compiler construction [1] where they are used for reducing execution time. The idea of using liveness analysis techniques for state space reduction was first described by Bozga *et al.* [3]. In [13] a similar technique was presented, using an analysis of the control flow graph.

The aforementioned techniques are restricted to an analysis of state spaces. In [9] a number of static analysis techniques, inspired by [4], were developed for PBESs. In *ibid.* the authors also showed that intractable verification problems can become tractable because of their static analysis techniques. Our methods generalise these techniques.

Structure of the paper. In Section 2 we give a cursory overview of basic PBES theory. In Section 3 we describe our construction of control flow graphs for PBESs. These are used in Section 4 to determine live variables and reset irrelevant parameters. We present an optimisation of the analysis in Section 5. The approach is evaluated in Section 6, and we conclude in Section 7.

2 Preliminaries

Throughout this paper, we work in a setting of *abstract data types* with non-empty data sorts $D_1, D_2, \ldots$, and operations on these sorts, and a set $\mathcal{D}$ of sorted data variables. We write vectors in boldface, *e.g.* $\boldsymbol{d}$ is used to denote a vector of data variables. We write $\boldsymbol{d}[i]$ to denote the i-th element of a vector $\boldsymbol{d}$.

A semantic set $\mathbb{D}$ is associated to every sort D, such that every term of sort D, and all operations on D are mapped to the elements and operations of $\mathbb{D}$ they represent. We assume an interpretation function $[\![\_]\!]$ that maps every closed term t of sort D to the data element $[\![t]\!]$ that it represents. For open terms we use an environment δ that maps each variable from $\mathcal{D}$ to a value of the associated type. The interpretation $[\![t]\!]\delta$ of an open term is given by $\delta(t)$, where the extension of δ to open terms is standard. Environment updates are denoted $\delta[v/d]$, where $\delta[v/d](d') = v$ if $d' = d$, and $\delta(d')$ otherwise.

We specifically assume the existence of a sort B with elements true and false representing the Booleans $\mathbb{B}$ and a sort $N = \{0, 1, 2, \ldots\}$ representing the natural numbers $\mathbb{N}$. For these sorts, we assume that the usual operators are available and, for readability, these are written the same as their semantic counterparts.

Before we formally define the notion of a parameterised Boolean equation system, we formalise the notion of *predicate formulae*.

Definition 1. Predicate formulae *are defined through the following grammar:*

$$\varphi, \psi ::= b \mid X(\boldsymbol{e}) \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \forall d\colon D.\varphi \mid \exists d\colon D.\varphi$$

in which b is a data term of sort B, X is a predicate variable of sort $\boldsymbol{D} \to B$, taken from some sufficiently large set $\mathcal{P}$ of predicate variables, and $\boldsymbol{e}$ is a vector of data terms of sort $\boldsymbol{D}$.

We assume that $\wedge$ and $\vee$ associate to the left, and that $\wedge$ binds stronger than $\vee$. Freely occurring data variables in φ are denoted by $FV(\varphi)$. Predicate formulae without predicate variables are called *simple*. We assume that if a data variable is bound by a quantifier in a formula φ, it does not also occur free within φ.

Definition 2. *The interpretation of a predicate formula φ in the context of a predicate environment $\eta\colon \mathcal{P} \to \mathbb{D} \to \mathbb{B}$ and data environment δ is denoted as $[\![\varphi]\!]\eta\delta$, where:*

$$\begin{aligned} [\![b]\!]\eta\delta &= [\![b]\!]\delta & [\![X(e)]\!]\eta\delta &= \eta(X)([\![e]\!]\delta) \\ [\![\varphi \wedge \psi]\!]\eta\delta &= [\![\varphi]\!]\eta\delta \wedge [\![\psi]\!]\eta\delta & [\![\varphi \vee \psi]\!]\eta\delta &= [\![\varphi]\!]\eta\delta \vee [\![\psi]\!]\eta\delta \\ [\![\forall d\colon D.\varphi]\!]\eta\delta &= \forall v \in \mathbb{D}.[\![\varphi]\!]\eta\delta[v/d] & [\![\exists d\colon D.\varphi]\!]\eta\delta &= \exists v \in \mathbb{D}.[\![\varphi]\!]\eta\delta[v/d] \end{aligned}$$

We define *logical equivalence* between two predicate formulae φ, ψ, denoted $\varphi \equiv \psi$, as $[\![\varphi]\!]\eta\delta = [\![\psi]\!]\eta\delta$ for all η, δ.

Parameterised Boolean equation systems (PBESs) are sequences of fixed point equations ranging over predicate formulae.

Definition 3. *PBESs are defined by the following grammar:*

$$\mathcal{E} ::= \epsilon \mid (\nu X(\boldsymbol{d} \colon \boldsymbol{D}) = \varphi)\mathcal{E} \mid (\mu X(\boldsymbol{d} \colon \boldsymbol{D}) = \varphi)\mathcal{E}$$

in which ϵ denotes the empty equation system; μ and ν are the least and greatest fixed point signs, respectively; X is a sorted predicate variable of sort $\boldsymbol{D} \to B$, $\boldsymbol{d}$ is a vector of formal parameters, and φ is a predicate formula. We henceforth omit a trailing ϵ.

Typically, φ_X denotes the right-hand side of the defining equation for X in a PBES $\mathcal{E}$. The set of *formal parameters* of a predicate variable X is denoted $\mathsf{par}(X)$; we assume that $FV(\varphi_X) \subseteq \mathsf{par}(X)$. Sometimes we superscript a formal parameter with the predicate variable it belongs to, *i.e.*, we may write d^X when $d \in \mathsf{par}(X)$.

Let $\mathsf{bnd}(\mathcal{E})$ denote the set of predicate variables occurring at the left hand sides of the equations in $\mathcal{E}$; these are $\mathcal{E}$'s *bound predicate variables*. Throughout this paper, we deal with PBESs that are both *well-formed* and *closed*: every bound predicate variable occurs in the left hand side of exactly one equation of $\mathcal{E}$, and all predicate variables occurring at the right-hand side are taken from $\mathsf{bnd}(\mathcal{E})$, respectively.

To each PBES $\mathcal{E}$ we associate a *top assertion*, denoted $\mathbf{init}\ X(\boldsymbol{v})$, where we require $X \in \mathsf{bnd}(\mathcal{E})$. For a parameter $\boldsymbol{d}[m] \in \mathsf{par}(X)$ for the top assertion $\mathbf{init}\ X(\boldsymbol{v})$ we define the value $\mathsf{init}(\boldsymbol{d}[m])$ as $\boldsymbol{v}[m]$.

We next define a PBES's semantics. Let $\mathbb{B}^{\mathbb{D}}$ denote the set of functions $f \colon \mathbb{D} \to \mathbb{B}$, and consider the ordering $\sqsubseteq$ on its elements, defined as $f \sqsubseteq g$ iff for all $v \in \mathbb{D}$, $f(v)$ implies $g(v)$. Observe that $(\mathbb{B}^{\mathbb{D}}, \sqsubseteq)$ is a complete lattice.

A single equation gives rise to a predicate transformer on this lattice as follows: a predicate formula φ can be viewed (syntactically) as a functional $\lambda \boldsymbol{d} \colon \boldsymbol{D}.\varphi$. The interpretation of $\lambda \boldsymbol{d} \colon \boldsymbol{D}.\varphi$, denoted $[\![\lambda \boldsymbol{d} \colon \boldsymbol{D}.\varphi]\!]\eta\delta$, is the functional $(\lambda \boldsymbol{v} \in \mathbb{D}.[\![\varphi]\!]\eta\delta[\boldsymbol{v}/\boldsymbol{d}])$, which is a function in $\mathbb{B}^{\mathbb{D}}$. The predicate transformer associated to this functional is given by $\lambda f \in \mathbb{B}^{\mathbb{D}}.[\![\lambda \boldsymbol{d} \colon \boldsymbol{D}.\varphi]\!]\eta[f/X]\delta$. Since the predicate transformers defined this way are monotonic and $(\mathbb{B}^{\mathbb{D}}, \sqsubseteq)$ is a complete lattice, the extremal fixed points of these predicate transformers exist. We denote these by $\sigma f \in \mathbb{B}^{\mathbb{D}}.[\![\lambda \boldsymbol{d} \colon \boldsymbol{D}.\varphi]\!]\eta[f/X]\delta$, for $\sigma \in \{\mu, \nu\}$. We now extend the semantics of individual equations to PBESs.

Definition 4. *The* solution *of an equation system in the context of a predicate environment η and data environment δ is defined inductively as follows:*

$$[\![\epsilon]\!]\eta\delta = \eta$$

$$[\![(\sigma X(\boldsymbol{d} \colon \boldsymbol{D}) = \varphi_X)\mathcal{E}]\!]\eta\delta = [\![\mathcal{E}]\!](\eta[\sigma f \in \mathbb{B}^{\mathbb{D}}.[\![\lambda \boldsymbol{d} \colon \boldsymbol{D}.\varphi_X]\!]([\![\mathcal{E}]\!]\eta[f/X]\delta)\delta/X])\delta$$

The solution prioritises the fixed point signs of equations that come first over the fixed point signs of equations that follow, while respecting the equations. The solution to a predicate variable in a *closed* PBES is independent of the predicate and data environments in which it is evaluated. We therefore typically leave out these environments and write $[\![\mathcal{E}]\!](X)$ instead of $[\![\mathcal{E}]\!]\eta\delta(X)$.

The *signature* [12] of a predicate variable X of sort $\boldsymbol{D} \to B$, $\mathsf{sgt}(X)$, is the product $\{X\} \times \mathbb{D}$. The notion of signature is lifted to sets of predicate variables $P \subseteq \mathcal{P}$ in the natural way, *i.e.* $\mathsf{sgt}(P) = \bigcup_{X \in P} \mathsf{sgt}(X)$.[1]

Definition 5 ([12, Definition 6]). *Let* $R \subseteq \mathsf{sgt}(\mathcal{P}) \times \mathsf{sgt}(\mathcal{P})$ *be an arbitrary relation. A predicate environment* η *is an* R*-correlation iff* $(X, \boldsymbol{v})\ R\ (X', \boldsymbol{v}')$ *implies* $\eta(X)(\boldsymbol{v}) = \eta(X')(\boldsymbol{v}')$.

A *block* is a non-empty equation system of like-signed fixed point equations. Given an equation system $\mathcal{E}$, a block $\mathcal{B}$ is maximal if its neighbouring equations in $\mathcal{E}$ are of a different sign than the equations in $\mathcal{B}$. The i^{th} maximal block in $\mathcal{E}$ is denoted by $\mathcal{E}\rceil i$. For relations R we write Θ_R for the set of R-correlations.

Definition 6 ([12, Definition 7]). *Let* $\mathcal{E}$ *be an equation system. A relation* $R \subseteq \mathsf{sgt}(\mathcal{P}) \times \mathsf{sgt}(\mathcal{P})$ *is a* consistent correlation *on* $\mathcal{E}$*, if for* $X, X' \in \mathsf{bnd}(\mathcal{E})$, $(X, \boldsymbol{v})\ R\ (X', \boldsymbol{v}')$ *implies:*

1. *for all* i, $X \in \mathsf{bnd}(\mathcal{E}\rceil i)$ *iff* $X' \in \mathsf{bnd}(\mathcal{E}\rceil i)$
2. *for all* $\eta \in \Theta_R$, δ, *we have* $[\![\varphi_X]\!]\eta\delta[\boldsymbol{v}/\boldsymbol{d}] = [\![\varphi'_X]\!]\eta\delta[\boldsymbol{v}'/\boldsymbol{d}']$

For $X, X' \in \mathsf{bnd}(\mathcal{E})$, *we say* $(X, \boldsymbol{v})$ *and* $(X', \boldsymbol{v}')$ *consistently correlate, denoted as* $(X, \boldsymbol{v}) \doteqdot (X', \boldsymbol{v}')$ *iff there exists a correlation* $R \subseteq \mathsf{sgt}(\mathsf{bnd}(\mathcal{E})) \times \mathsf{sgt}(\mathsf{bnd}(\mathcal{E}))$ *such that* $(X, \boldsymbol{v})\ R\ (X', \boldsymbol{v}')$.

Consistent correlations can be lifted to variables in different equation systems in $\mathcal{E}$ and $\mathcal{E}'$, assuming that the variables in the equation systems do not overlap. We call such equation systems *compatible*. Lifting consistent correlations to different equation system can, *e.g.*, be achieved by merging the equation systems to an equation system $\mathcal{F}$, in which, if $X \in \mathsf{bnd}(\mathcal{E})$, then $X \in \mathsf{bnd}(\mathcal{E}\rceil i)$ iff $X \in \mathsf{bnd}(\mathcal{F}\rceil i)$, and likewise for $\mathcal{E}'$. The consistent correlation can then be defined on $\mathcal{F}$.

The following theorem [12] shows the relation between consistent correlations and the solution of a PBES.

Theorem 1 ([12, Theorem 2]). *Let* $\mathcal{E}$, $\mathcal{E}'$ *be compatible equation systems, and* $\doteqdot$ *a consistent correlation. Then for all* $X \in \mathsf{bnd}(\mathcal{E})$, $X' \in \mathsf{bnd}(\mathcal{E}')$ *and all* $\eta \in \Theta_{\doteqdot}$, *we have* $(X, \boldsymbol{v}) \doteqdot (X', \boldsymbol{v}') \implies [\![\mathcal{E}]\!]\eta\delta(X)(\boldsymbol{v}) = [\![\mathcal{E}']\!]\eta\delta(X')(\boldsymbol{v}')$

We use this theorem in proving the correctness of our static analysis technique.

PBESs can be obtained from a variety of verification problems. To solve a PBES, and thereby the verification problem it encodes, it is typically *instantiated* [10] into a *Boolean equation system* (BES), an equation system without data and quantification, using a process similar to explicit state space generation, and for which solving is decidable. Reducing the time spent on instantiation is therefore instrumental in speeding up solving PBESs. The example below, used as a running example throughout the paper, illustrates how a model checking problem can be reduced to a PBES solving problem.

[1] Note that in [12] the notation sig is used to denote the signature. Here we deviate from this notation due to the naming conflict with the *significant parameters* of a formula, which also is standard notation introduced in [9], and which we introduce in Section 4.

Example 1. Consider the following specification of a lossy one place buffer that, when $s = 1$, can read a data element through $receive$, and then, non-deterministically (by means of the τ transitions), loses the data element when $s = 3$, or forwards the data element through $send$ when $s = 4$. After this, it is back in its initial state. For messages we use a type D, containing at least the element d_1.

$$
\begin{aligned}
&\mathbf{proc}\ P(s\colon N, d\colon D) = \textstyle\sum_{e\colon D}(s = 1) \to \mathit{receive}(e).P(2, e)\\
&\quad +(s = 2) \to \tau.P(3, d) + (s = 2) \to \tau.P(4, d)\\
&\quad +(s = 3) \to \mathit{lost}.P(1, d) + (s = 4) \to \mathit{send}(d).P(1, d_1);\\
&\mathbf{init}\ P(1, d_1);
\end{aligned}
$$

The (first order) modal μ-calculus formula below asserts that invariantly, if a message v is received through $receive$, then, as long as no other message is read through $receive$, all messages delivered must match message v.

$$
\begin{aligned}
&\nu X.[\mathit{true}]X \land\\
&\qquad (\forall v\colon D.[\mathit{receive}(v)]\nu Y.([\overline{\exists w\colon D.\mathit{receive}(w)}]Y \land \forall u\colon D.[\mathit{send}(u)](v = u))).
\end{aligned}
$$

The model checking problem whether the lossy buffer satisfies the above formula is converted to the following PBES. Observe that in this PBES, the equation for X depends on that of Y: in the first conjunct of the equation for X, there is a recursion to the equation for Y through $Y(2, e, e)$.

$$
\begin{aligned}
&\nu X(s\colon N, d\colon D) = (\forall e\colon D.s = 1 \implies Y(2, e, e)) \land (\forall e\colon D.s = 1 \implies X(2, e))\\
&\quad \land(s = 2 \implies X(3, d)) \land (s = 2 \implies X(4, d)) \land (s = 3 \implies X(1, d))\\
&\quad \land(s = 4 \implies X(1, d_1))\\
&\mu Y(s\colon N, d, v\colon D) = (s = 4 \implies d = v) \land (s = 2 \implies Y(3, d, v))\\
&\quad \land(s = 2 \implies Y(4, d, v)) \land (s = 3 \implies Y(1, d, v)) \land (s = 4 \implies Y(1, d_1, v))\\
&\mathbf{init}\ X(1, d_1);
\end{aligned}
$$

3 Reconstructing control flow

Our static analysis techniques presented in the next sections are based on a notion of *control flow* in a PBES. In contrast to a setting in which the artefacts analysed have a clear graph structure, and for which the notion of control flow is more or less a commonly understood concept, in our setting of PBESs, this is not the case. This is largely due to the fact that a PBES consists of sequences of equations over predicate formulae, for which there is no obvious graph structure.

In Section 3.1, we propose a notion of *control flow parameters* that permits us to define a control flow graph that is meaningful in our setting in Section 3.2. We describe heuristics for computing control flow graphs efficiently in Section 3.3.

3.1 Control Flow Parameters

The parameters of an equation in a PBES typically encode (parts of) the state space of a system and the information vital for the property that is verified in a model checking

problem. Similarly, in the encoding of equivalence checking problems, the parameters of a PBES typically encode (parts of) the state spaces of both systems that are compared. It is therefore to be expected that the control of a system is reflected by the changes of values of a subset of the parameters in a PBES equation. The predicate variable instances of the form $X(\boldsymbol{e})$, present in the right-hand sides of the equations in a PBES, essentially dictate how the values of the parameters change.

In view of these observations, we are interested in identifying how the predicate variable instances affect the values of parameters. A complication is that there can be many different occurrences of syntactically indistinguishable predicate variable instances that, due to the context in which they are contained in a predicate formula, can be semantically different. We therefore first introduce notation to identify individual predicate variable instances in a formula.

We denote the number of predicate variable instances occurring in a predicate formula φ by $\mathsf{npred}(\varphi)$. We assume that predicate variable instances in φ are assigned a unique natural number between 1 and $\mathsf{npred}(\varphi)$.

Definition 7. *Let φ be a predicate formula and let i be between 1 and $\mathsf{npred}(\varphi)$. The functions $\mathsf{pred}(\varphi, i)$, $\mathsf{data}(\varphi, i)$ and $\mathsf{PVI}(\varphi, i)$ are such that the predicate variable instance $\mathsf{PVI}(\varphi, i)$ is the i^{th} predicate variable instance in φ, syntactically present as $\mathsf{pred}(\varphi, i)(\mathsf{data}(\varphi, i))$.*

We define the syntactic replacement of the predicate variable instance at position i by ψ in formula φ, denoted as $\varphi[i \mapsto \psi]$, as follows.

Definition 8. *Let ψ be a predicate formula, and let $i \leqslant \mathsf{npred}(\varphi)$, $\varphi[i \mapsto \psi]$ is defined inductively as follows.*

$$
\begin{aligned}
b[i \mapsto \psi] &= b \\
Y(e)[i \mapsto \psi] &= \begin{cases} \psi & \textit{if } i = 1 \\ Y(e) & \textit{otherwise} \end{cases} \\
(\forall d\colon D.\varphi)[i \mapsto \psi] &= \forall d\colon D.\varphi[i \mapsto \psi] \\
(\exists d\colon D.\varphi)[i \mapsto \psi] &= \exists d\colon D.\varphi[i \mapsto \psi] \\
(\varphi_1 \wedge \varphi_2)[i \mapsto \psi] &= \begin{cases} \varphi_1 \wedge \varphi_2[(i - \mathsf{npred}(\varphi_1)) \mapsto \psi] & \textit{if } i > \mathsf{npred}(\varphi_1) \\ \varphi_1[i \mapsto \psi] \wedge \varphi_2 & \textit{if } i \leqslant \mathsf{npred}(\varphi_1) \end{cases} \\
(\varphi_1 \vee \varphi_2)[i \mapsto \psi] &= \begin{cases} \varphi_1 \vee \varphi_2[(i - \mathsf{npred}(\varphi_1)) \mapsto \psi] & \textit{if } i > \mathsf{npred}(\varphi_1) \\ \varphi_1[i \mapsto \psi] \vee \varphi_2 & \textit{if } i \leqslant \mathsf{npred}(\varphi_1) \end{cases}
\end{aligned}
$$

A *control flow parameter* is, intuitively, a parameter whose exact value we always know *before* and *after* recursing via a predicate variable instance. That is, we require of a control flow parameter that a recursion through the predicate variable instance is possible only when the control flow parameter has a fixed, known value, and, at the same time, we know the effect this recursion has on the value of the control flow parameter. We now make this idea more precise, by employing a collection of partial functions.

Definition 9. *Let* $s\colon \mathcal{P} \times \mathbb{N} \times \mathbb{N} \to D$, $c\colon \mathcal{P} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *and* $t\colon \mathcal{P} \times \mathbb{N} \times \mathbb{N} \to D$ *be partial functions, where* D *is the union of all ground data sort expressions. The triple* (s, t, c) *is a* unicity constraint *for PBES* $\mathcal{E}$ *if, for all* $X \in \mathsf{bnd}(\mathcal{E})$ *and* $1 \leqslant i \leqslant \mathsf{npred}(\varphi_X)$*:*

- *if* $s(X,i,j){=}e$ *then* $\varphi_X \equiv \varphi_X[i \mapsto (\boldsymbol{d}[j] = e \wedge \mathsf{PVI}(\varphi_X, i))]$,
- *if* $t(X,i,j){=}e$ *then* $\varphi_X \equiv \varphi_X[i \mapsto (\mathsf{data}(\varphi_X, i)[j] = e \wedge \mathsf{PVI}(\varphi_X, i))]$,
- *if* $c(X,i,j){=}k$ *then* $\varphi_X \equiv \varphi_X[i \mapsto (\mathsf{data}(\varphi_X, i)[k] = \boldsymbol{d}[j] \wedge \mathsf{PVI}(\varphi_X, i))]$.

The function s in a unicity constraint exactly captures that, when defined for $s(X, i, j)$, the i^{th} predicate variable instance in φ_X only needs to be considered in case variable $\boldsymbol{d}[j]$ has the value $s(X, i, j)$. In particular, for any other value of the variable $\boldsymbol{d}[j]$, the truth of the predicate variable instance is immaterial to the truth of φ_X. In the same vein, when $t(X, i, j)$ is defined, then the j^{th} data expression that is an argument to the i^{th} predicate variable instance in φ_X has a fixed value, given by $t(X, i, j)$. The function c allows us to establish that, whenever $c(X, i, j)$ is defined to be k, then the value of variable $\boldsymbol{d}[j]$ is copied to the data expression on position k in the i^{th} predicate variable instance of φ_X. Note that whenever a function is not defined (denoted by $\perp$), we can draw no meaningful information from that.

Example 2. Reconsider Example 1. The triple (s, t, c), where $s(X, 1, 1) = s(X, 2, 1) = 1$, $t(X, 1, 1) = 2$, $t(X, 3, 1) = 3$ and $c(X, 4, 2) = 2$ is a unicity constraint. By extending the mapping c by defining $c(X, 5, 2) = 2$, (s, t, c) remains a unicity constraint, but $c(X, 6, 2)$ may only be defined when D contains only elements equal to d_1.

The requirements allow unicity constraints to be underspecified. In practice, it is desirable to choose the constraints as complete as possible. If, in a unicity constraint (s, t, c), s and c are defined for a predicate variable instance, it can immediately be established that we can define t as well. This is formalised by the following property.

Property 1. Let X be a predicate variable, $i \leqslant \mathsf{npred}(\varphi_X)$, let (s, t, c) be a unicity constraint, and let e be a value, then

$$(s(X, i, n) = e \wedge c(X, i, n) = m) \implies t(X, i, m) = e.$$

Henceforth we assume that all unicity constraints satisfy this property. The overlap between t and c is now straightforwardly formalised in the following lemma.

Lemma 1. *Let* X *be a predicate variable,* $i \leqslant \mathsf{npred}(\varphi_X)$*, and let* (s, t, c) *be a unicity constraint, then if* $(s(X, i, n)$ *and* $t(X, i, m)$ *are both defined,*

$$c(X, i, n) = m \implies s(X, i, n) = t(X, i, m).$$

Proof. Immediately from the definitions and Property 1.

We next —incrementally— characterise a meaningful set of control flow parameters by imposing restrictions on the unicity constraints. The first restriction —a local restriction— is such that any parameter that qualifies as a control flow parameter only affects itself and is affected by itself through self-recursions.

Definition 10. *A parameter* $d[n] \in \mathsf{par}(X)$ *is a* local control flow parameter *(LCFP) if for all* i *such that* $\mathsf{pred}(\varphi_X, i) = X$*, either* $\mathsf{source}(X, i, n)$ *and* $\mathsf{dest}(X, i, n)$ *are defined, or* $\mathsf{copy}(X, i, n) = n$*.*

The second restriction —a global restriction— is such that *if* a potential control flow parameter is affected by a parameter in another equation, then this can only be because the other parameter is also a potential control flow parameter, and the recursion copies the value of this potential control flow parameter.

Definition 11. *LCFP* $d[n] \in \mathsf{par}(X)$ *is a* global control flow parameter *(GCFP) if for all* $Y \in \mathsf{bnd}(\mathcal{E})\backslash\{X\}$ *and all* i *such that* $\mathsf{pred}(\varphi_Y, i) = X$*, either* $\mathsf{dest}(Y, i, n)$ *is defined, or* $\mathsf{copy}(Y, i, m) = n$ *for some GCFP* $d[m] \in \mathsf{par}(Y)$*.*

It may still be that, *e.g.*, a GCFP affects multiple GCFPs in one other equation. Through transitivity, this can lead to a GCFP in an equation affecting another GCFP in the *same* equation; this is conceptually quite unnatural and therefore undesirable. We impose further restrictions on GCFPs, ensuring that ultimately, a control flow parameter affects at most one other control flow parameter in each other equation. Such control flow parameters are considered to be "identical". We say that GCFPs $d^X[n]$ and $d^Y[m]$ are *related*, denoted $d^X[n] \sim d^Y[m]$, if $n = \mathsf{copy}(Y, i, m)$ for some i. Let $\sim^*$ denote the reflexive, symmetric and transitive closure of $\sim$.

Definition 12. *Let* $\mathcal{C}$ *be a (sub)set of GCFPs and let* $\approx$ *be some equivalence relation on* $\mathcal{C}$ *satisfying* $\sim^* \subseteq \approx$*. Then the pair* $\langle \mathcal{C}, \approx \rangle$ *defines a set of* control flow parameters (CFPs) *if for all* $X \in \mathsf{bnd}(\mathcal{E})$ *and all* $d, d' \in \mathcal{C} \cap \mathsf{par}(X)$*, if* $d \approx d'$*, then* $d = d'$*.*

A unicity constraint is a *witness* to a set of control flow parameters $\langle \mathcal{C}, \approx \rangle$ if the unicity constraint induces a pair $\langle \mathcal{C}, \approx \rangle$ through Definition 12.

We say that two parameters $c, c' \in \mathcal{C}$ are *identical control flow parameters* if $c \approx c'$. It is easy to see that using the relation $\approx$, we can, across all equations in a PBES, assign a single unique name to identical control flow parameters. Control flow parameters that do not appear in an equation can easily be added to these without fundamentally changing their solution. Without loss of generality and for ease of readability we henceforth work under the following assumption.

Assumption 1 *The set of control flow parameters is the same for every equation in a PBES; that is, for all* $X, Y \in \mathsf{bnd}(\mathcal{E})$ *in a PBES* $\mathcal{E}$ *we have* $d \in \mathsf{par}(X)$ *is a CFP iff* $d \in \mathsf{par}(Y)$ *is a CFP, and both parameters are identical control flow parameters.*

Example 3. The unicity constraint of Example 2 can be extended in such a way that the parameter s of equation X (denoted s^X) and parameter s of equation Y (denoted s^Y) satisfy all requirements of CFP defined by the pair $\langle \{s^X, s^Y\}, \approx \rangle$, where $\approx$ is the smallest equivalence relation such that $s^X \approx s^Y$. Hence, both parameter s^X and parameter s^Y are control flow parameters. Both already share the same name, so assumption 1 is already met. Note that there is no unicity constraint that enables us to mark parameters d^X, v^Y and d^Y as control flow parameters.

Henceforth, any parameter that is not a control flow parameter is called a *data parameter*. For ease of reasoning, we make this distinction explicit by partitioning $\mathcal{D}$ into

$\mathcal{D}^{CFP}$ and $\mathcal{D}^{DP}$, containing the control flow parameters and the data parameters respectively. We occasionally write equations as $\sigma X(\boldsymbol{c} \colon \boldsymbol{C}, \boldsymbol{d}^X : \boldsymbol{D}^X) = \varphi_X(\boldsymbol{c})(\boldsymbol{d}^X)$, where $\boldsymbol{c}$ are the CFPs, and $\boldsymbol{d}^X$ are the DPs of the equation for X. Observe that $\boldsymbol{c}$ is not superscripted (in line with Assumption 1). If the equation X is clear from the context, we also omit the superscript of $\boldsymbol{d}$.

3.2 Control Flow Graph

From hereon, let $\mathcal{E}$ be an arbitrary PBES and let $\boldsymbol{c}$ be a vector of control flow parameters of $\mathcal{E}$, witnessed by unicity constraint $(\mathsf{source}, \mathsf{dest}, \mathsf{copy})$. We next construct a control flow graph that describes how the values of the control flow parameters are affected by the predicate variable instances that occur in the predicate formulae of the PBES. The edge relation is determined using the unicity constraint $(\mathsf{source}, \mathsf{dest}, \mathsf{copy})$ that witnesses $\boldsymbol{c}$. The locations in the graph are valuations for the control flow parameters.

Definition 13. *The set of values* $\boldsymbol{c}[k]$ *can attain, denoted* $\mathsf{values}(\boldsymbol{c}[k])$*, is defined as:* $\{\mathsf{init}(\boldsymbol{c}[k])\} \cup \{v \mid \exists i \in \mathbb{N} : \exists X \in \mathsf{bnd}(\mathcal{E}) : \mathsf{source}(X, i, k) = v \vee \mathsf{dest}(X, i, k) = v\}$.

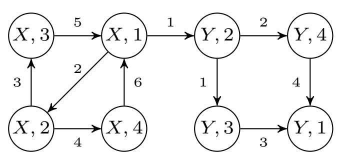
Note that the set $\mathsf{values}(\boldsymbol{c}[k])$ is finite. We generalise values to the vector $\boldsymbol{c}$ in the obvious way. The control flow graph of $\mathcal{E}$ is constructed as follows.

Definition 14. *The control flow graph (CFG) of* $\mathcal{E}$ *is a graph* $(V, \to)$ *with:*

- $V = \mathsf{bnd}(\mathcal{E}) \times \mathsf{values}(\boldsymbol{c})$*, and*
- $\to \subseteq V \times \mathbb{N} \times V$ *is the least relation for which, whenever* $(X, \boldsymbol{v}) \xrightarrow{i} (\mathsf{pred}(i), \boldsymbol{w})$ *then for every* k *either:*
  - $\mathsf{source}(X, i, k) = \boldsymbol{v}[k]$ *and* $\mathsf{dest}(X, i, k) = \boldsymbol{w}[k]$*, or*
  - $\mathsf{source}(X, i, k) = \bot$, $\mathsf{copy}(X, i, k) = k$ *and* $\boldsymbol{v}[k] = \boldsymbol{w}[k]$*, or*
  - $\mathsf{source}(X, i, k) = \bot$*, and* $\mathsf{dest}(X, i, k) = \boldsymbol{w}[k]$.

Henceforth, we refer to the vertices in the control flow graph as control flow locations, or *locations*, for short.

Example 4. Reconsider the PBES from Example 1 and its control flow parameter (parameter s), determined in Example 3. Its control flow graph is depicted below.



Control flow graphs are complete in the sense that all predicate variable instances potentially influencing the truth of φ_X at location $\boldsymbol{v}$ are neighbours of the vertex $(X, \boldsymbol{v})$. Instances that cannot affect its truth are not.

Lemma 2. *Let* $(V, \to)$ *be* $\mathcal{E}$*'s control flow graph. Then for all* $(X, \boldsymbol{v}) \in V$ *and all predicate environments* η, η' *and data environments* δ*:*

$$[\![\varphi_X]\!]\eta\delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}] = [\![\varphi_X]\!]\eta'\delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}]$$

provided that $\eta(Y)(\boldsymbol{w}) = \eta'(Y)(\boldsymbol{w})$ *for all* $(Y, \boldsymbol{w})$ *satisfying* $(X, \boldsymbol{v}) \xrightarrow{i} (Y, \boldsymbol{w})$.

Proof. Let η, η' be predicate environments and δ be a data environment, and let $(X, \boldsymbol{v}) \in V$. Suppose that for all $(Y, \boldsymbol{w})$ for which $(X, \boldsymbol{v}) \xrightarrow{i} (Y, \boldsymbol{w})$, we know that $\eta(Y)(\boldsymbol{w}) = \eta'(Y)(\boldsymbol{w})$.

Towards a contradiction, let $[\![\varphi_X]\!]\eta\delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}] \neq [\![\varphi_X]\!]\eta'\delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}]$. Then there must be a predicate variable instance $\mathsf{PVI}(\varphi_X, i,)$ such that

$$\begin{aligned} &\eta(\mathsf{pred}(\varphi_X, i))([\![\mathsf{data}(\varphi_X, i)]\!]\delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}]) \\ \neq\ &\eta'(\mathsf{pred}(\varphi_X, i))([\![\mathsf{data}(\varphi_X, i)]\!]\delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}]). \end{aligned} \tag{1}$$

Let $\mathsf{data}(\varphi_X, i) = (\boldsymbol{e}, \boldsymbol{e}')$, where $\boldsymbol{e}$ are the values of the control flow parameters, and $\boldsymbol{e}'$ are the values of the data parameters.

Consider an arbitrary control flow parameter $\boldsymbol{c}[\ell]$. We distinguish two cases:

- $\mathsf{source}(X, i, \ell) \neq \bot$. Then we know $\mathsf{dest}(X, i, \ell) \neq \bot$, and the requirement for the edge $(X, \boldsymbol{v}) \xrightarrow{i} (\mathsf{pred}(\varphi_X, i), \boldsymbol{e})$ is satisfied for ℓ.
- $\mathsf{source}(X, i, \ell) = \bot$. Since $\boldsymbol{c}[\ell]$ is a control flow parameter, we can distinguish two cases based on Definitions 11 and 12:
  - $\mathsf{dest}(X, i, \ell) \neq \bot$. Then parameter ℓ immediately satisfies the requirements that show the existence of the edge $(X, \boldsymbol{v}) \xrightarrow{i} (\mathsf{pred}(\varphi_X, i), \boldsymbol{e})$ in the third clause in the definition of CFG.
  - $\mathsf{copy}(X, i, \ell) = \ell$. According to the definition of copy, we now know that $\boldsymbol{v}[\ell] = \boldsymbol{e}[\ell]$, hence the edge $(X, \boldsymbol{v}) \xrightarrow{i} (\mathsf{pred}(\varphi_X, i), \boldsymbol{e})$ exists according to the second requirement in the definition of CFG.

Since we have considered an arbitrary ℓ, we know that for all ℓ the requirements are satisfied, hence $(X, \boldsymbol{v}) \xrightarrow{i} (\mathsf{pred}(\varphi_X, i), \boldsymbol{e})$. Then according to the definition of η and η', $\eta(\mathsf{pred}(\varphi_X, i))([\![\boldsymbol{e}]\!]\delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}]) = \eta'(\mathsf{pred}(\varphi_X, i))([\![\boldsymbol{e}]\!]\delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}])$. This contradicts (1), hence we find that $[\![\varphi_X]\!]\eta\delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}] = [\![\varphi_X]\!]\eta'\delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}]$.

3.3 Heuristics

The unicity constraints and the control flow parameters and graph they induce, as presented in the previous sections, are not necessarily efficiently or effectively computable. We therefore look for cheap heuristics that permit us to determine these unicity constraints. For this, it suffices to approximate which subformulae in a predicate formula are required to be true for a predicate variable instance to still be relevant to the truth of the predicate formula. From such subformulae, referred to as *guards*, we subsequently heuristically determine a good unicity constraint. Guards are defined as follows.

Definition 15. *Let φ be a predicate formula. We define the* guard *of predicate variable instance* $\mathsf{PVI}(\varphi, i)$ *for* $i \leqslant \mathsf{npred}(\varphi)$ *inductively as follows:*

$$\begin{aligned}
\mathsf{guard}^i(b) &= \mathit{false} \\
\mathsf{guard}^i(Y) &= \mathit{true} \\
\mathsf{guard}^i(\forall d\colon D.\varphi) &= \mathsf{guard}^i(\varphi) \\
\mathsf{guard}^i(\exists d\colon D.\varphi) &= \mathsf{guard}^i(\varphi) \\
\mathsf{guard}^i(\varphi \wedge \psi) &= \begin{cases} s(\varphi) \wedge \mathsf{guard}^{i-\mathsf{npred}(\varphi)}(\psi) & \textit{if } i > \mathsf{npred}(\varphi) \\ s(\psi) \wedge \mathsf{guard}^i(\varphi) & \textit{if } i \leqslant \mathsf{npred}(\varphi) \end{cases} \\
\mathsf{guard}^i(\varphi \vee \psi) &= \begin{cases} ns(\varphi) \wedge \mathsf{guard}^{i-\mathsf{npred}(\varphi)}(\psi) & \textit{if } i > \mathsf{npred}(\varphi) \\ ns(\psi) \wedge \mathsf{guard}^i(\varphi) & \textit{if } i \leqslant \mathsf{npred}(\varphi) \end{cases}
\end{aligned}$$

where

$$s(\varphi) = \begin{cases} \varphi & \textit{if } \mathsf{npred}(\varphi) = 0 \\ \mathit{true} & \textit{otherwise} \end{cases} \qquad ns(\varphi) = \begin{cases} \neg\varphi & \textit{if } \mathsf{npred}(\varphi) = 0 \\ \mathit{true} & \textit{otherwise} \end{cases}$$

Intuitively, the truth of a predicate variable instance $\mathsf{PVI}(\varphi, i)$ in a formula φ is irrelevant if $\mathsf{guard}^i(\varphi)$ is unsatisfiable. To show that, indeed, we compute a guard, we first show that we can guard every predicate variable instance with its guard, without changing the solution.

Lemma 3. *Let φ be a predicate formula, and let $i \leqslant \mathsf{npred}(\varphi)$, then for every predicate environment η and data environment δ,*

$$[\![\varphi]\!]\eta\delta = [\![\varphi[i \mapsto (\mathsf{guard}^i(\varphi) \implies \mathsf{PVI}(\varphi, i))]]\!]\eta\delta.$$

Proof. Let η and δ be arbitrary. We proceed by induction on φ. The base cases where $\varphi = b$ and $\varphi = Y(\boldsymbol{e})$ are trivial, and $\forall d\colon D.\psi$ and $\exists d\colon D.\psi$ follow immediately from the induction hypothesis. We describe the case where $\varphi = \varphi_1 \wedge \varphi_2$ in detail, the $\varphi = \varphi_1 \vee \varphi_2$ is completely analogous.

Assume that $\varphi = \varphi_1 \wedge \varphi_2$. Let $i \leqslant \mathsf{npred}(\varphi_1 \wedge \varphi_2)$. Without loss of generality assume that $i \leqslant \mathsf{npred}(\varphi_1)$, the other case is analogous. According to the induction hypothesis,

$$[\![\varphi_1]\!]\eta\delta = [\![\varphi_1[i \mapsto (\mathsf{guard}^i(\varphi_1) \implies \mathsf{PVI}(\varphi_1, i))]]\!]\eta\delta \tag{2}$$

We distinguish two cases.

- $\mathsf{occ}(\varphi_2) \neq \emptyset$. Then $[\![\mathsf{guard}^i(\varphi_1)]\!]\delta\eta = [\![\mathsf{guard}^i(\varphi_1 \wedge \varphi_2)]\!]\delta\eta$ according to the definition of guard. Since $i \leqslant \mathsf{npred}(\varphi_1)$, we find that $[\![\varphi_1 \wedge \varphi_2]\!]\eta\delta = [\![(\varphi_1 \wedge \varphi_2)[i \mapsto (\mathsf{guard}^i(\varphi_1 \wedge \varphi_2) \implies \mathsf{PVI}(\varphi_1 \wedge \varphi_2, i))]]\!]\eta\delta$.
- $\mathsf{occ}(\varphi_2) = \emptyset$. We have to show that

$$[\![\varphi_1 \wedge \varphi_2]\!]\eta\delta = [\![\varphi_1[i \mapsto (\mathsf{guard}^i(\varphi_1 \wedge \varphi_2) \implies \mathsf{PVI}(\varphi_1, i))] \wedge \varphi_2]\!]\eta\delta$$

From the semantics, it follows that $[\![\varphi_1 \wedge \varphi_2]\!]\eta\delta = [\![\varphi_1]\!]\eta\delta \wedge [\![\varphi_2]\!]\eta\delta$. Combined with (2), and an application of the semantics, this yields

$$[\![\varphi_1 \wedge \varphi_2]\!]\eta\delta = [\![\varphi_1[i \mapsto (\mathsf{guard}^i(\varphi_1) \implies \mathsf{PVI}(\varphi_1, i))] \wedge \varphi_2]\!]\eta\delta.$$

According to the definition of guard, $\mathsf{guard}^i(\varphi_1 \wedge \varphi_2) = \varphi_2 \wedge \mathsf{guard}^i(\varphi_1)$. Since φ_2 is present in the context, using monotonicity and an application of the semantics, the desired result follows. □

We can generalise the above, and guard every predicate variable instance in a formula with its guard, which preserves the solution of the formula. To this end we introduce the function guarded.

Definition 16. *Let φ be a predicate formula, then*

$$\mathsf{guarded}(\varphi) = \varphi[i \mapsto (\mathsf{guard}^i(\varphi) \implies \mathsf{PVI}(\varphi, i))]_{i \leqslant \mathsf{npred}(\varphi)}$$

where $[i \mapsto \psi_i]_{i \leqslant \mathsf{npred}(\varphi)}$ is the simultaneous syntactic substitution of all $\mathsf{PVI}(\varphi, i)$ with ψ_i.

The following corollary follows immediately from Lemma 3.

Corollary 1. *For all formulae φ, and for all predicate environments η, and data environments δ, $[\![\varphi]\!]\eta\delta = [\![\mathsf{guarded}(\varphi)]\!]\eta\delta$*

This corollary confirms our intuition that indeed the guards we compute effectively guard the recursions in a formula.

A good heuristic for defining the unicity constraints is by looking for positive occurrences of constraints of the form $d = e$ in the guards; these can be used to define the source function. For determining the dest function, one can replace a data parameter by the value dictated by source for this parameter for a predicate variable instance and check whether data expressions in a recursion reduce to a constant under this substitution. The copy function can be defined through simple syntactic checks that determine which data expressions in a predicate variable instance consist of data parameters only.

Once these functions are defined, the relation $\sim$ of related GCFPs can be computed. From this, the largest subset of GCFPs for which there is an equivalence relation $\approx$ of Definition 12 that subsumes $\sim^*$ can be determined by eliminating GCFPs that conflict with the requirement of Definition 12. Any subset of GCFPs, with induced relation $\sim^*$ on that subset that does not violate the requirement of Definition 12, already defines CFPs. A coarser relation $\approx \supseteq \sim^*$ can then be built by trying to relate GCFPs that, *e.g.*, share the same name, type or position across the parameter list of equations.

4 Data flow analysis

We now formalised the notion of a control flow parameter, and established heuristics to determine those parameters. Next we analyse the flow of data within an equation system. Intuitively, a data parameter d of X is potentially relevant if it can influence the

truth of φ_X. The influence is direct if d occurs in a Boolean expression in φ_X. Such parameters are called *significant* [9]; they can be determined as follows.

$$\begin{aligned} \mathsf{sig}(b) &= FV(b) & \mathsf{sig}(Y(e)) &= \varnothing \\ \mathsf{sig}(\varphi \wedge \psi) &= \mathsf{sig}(\varphi) \cup \mathsf{sig}(\psi) & \mathsf{sig}(\varphi \vee \psi) &= \mathsf{sig}(\varphi) \cup \mathsf{sig}(\psi) \\ \mathsf{sig}(\exists d\colon D.\varphi) &= \mathsf{sig}(\varphi) \backslash \{d\} & \mathsf{sig}(\forall d\colon D.\varphi) &= \mathsf{sig}(\varphi) \backslash \{d\} \end{aligned}$$

Suppose we have a function simplify that converts φ into an equivalent formula with fewer or equal numbers of significant parameters; *i.e.* $\mathsf{simplify}(\varphi) \equiv \varphi$, and $\mathsf{sig}(\varphi) \supseteq \mathsf{sig}(\mathsf{simplify}(\varphi))$. Typically simplify can be implemented by rewrite rules.

First, observe that if we assign values to some parameters, the likelihood that simplify can reduce the number of significant parameters in a formula increases. Second, observe that in a location of a control flow graph, we know the values for all control flow parameters. We can thus simplify formulae using these values. This will be the basis for our analysis. Then, through a backwards reachability using our control flow graph, we identify parameters that indirectly influence the values of significant parameters in the locations of the graph. Such indirect influences are the result of parameters affecting the value of other parameters through predicate variable instances.

Definition 17. *Let $\mathcal{E}$ be a PBES and let $(V, \rightarrow)$ be its control flow graph. We define marking $M : V \rightarrow \mathbb{P}(\mathcal{D}^{DP})$ inductively as follows:*

$$\begin{aligned} M^0(X, \boldsymbol{v}) &= \mathsf{sig}(\mathsf{simplify}(\varphi_X[\boldsymbol{c} := \boldsymbol{v}])) \\ M^{n+1}(X, \boldsymbol{v}) &= M^n(X, \boldsymbol{v}) \\ &\quad \cup \{d \in \mathsf{par}(X) \mid \exists i \in \mathbb{N}, (Y, \boldsymbol{w}) \in V : (X, \boldsymbol{v}) \xrightarrow{i} (Y, \boldsymbol{w}) \\ &\qquad \wedge \exists \boldsymbol{d}[\ell] \in M^n(Y, \boldsymbol{w}) :\ d \text{ affects } \mathsf{data}(\varphi_X, i)[\ell]\} \\ M(X, \boldsymbol{v}) &= \textstyle\bigcup_{n \in \mathbb{N}} M^n(X, \boldsymbol{v}) \end{aligned}$$

where variable d affects $\boldsymbol{e}[i]$ if $d \in FV(\boldsymbol{e}[i])$.

Note that the set M that is constructed approximates the set of parameters that are potentially relevant in a location. Parameters that do not end up in M for some location are guaranteed to be irrelevant.

Example 5. Analysing our running example using the control flow graph of Example 4 we find that, initially, data parameters d and v are marked in vertex $(Y, 4)$ only. In the next step the same parameters are marked in vertex $(Y, 2)$ due to the predicate variable instance at index 2. This is also the final marking.

The syntactic marking from Definition 17 induces a relation R^M on signatures as follows.

Definition 18. *Let $M : V \rightarrow \mathbb{P}(\mathcal{D}^{DP})$ be a marking. Every marking M induces a relation R^M such that $(X, [\![\boldsymbol{v}]\!], [\![\boldsymbol{w}]\!]) R^M (Y, [\![\boldsymbol{v}']\!], [\![\boldsymbol{w}']\!])$ if and only if $X = Y$, $[\![\boldsymbol{v}]\!] = [\![\boldsymbol{v}']\!]$, and $\forall \boldsymbol{d}[k] \in M(X, \boldsymbol{v}) : [\![\boldsymbol{w}[k]]\!] = [\![\boldsymbol{w}'[k]]\!]$.*

Observe that the relation R^M allows for relating *all* instances of the non-marked data parameters at a given control flow location. We prove that, if locations are related using

the relation R^M, then the corresponding instances in the PBES have the same solution by showing that R^M is a consistent correlation.

In order to prove this, we first show that given a predicate environment and two data environments, if the solution of a formula differs between those environments, and all predicate variable instances in the formula have the same solution, then there must be a significant parameter d in the formula that gets a different value in the two data environments.

Lemma 4. *For all formulae φ, predicate environments η, and data environments δ, δ', if we know that $[\![\varphi]\!]\eta\delta \neq [\![\varphi]\!]\eta\delta'$ and for all $i \leqslant \mathsf{npred}(\varphi)$, $[\![\mathsf{PVI}(\varphi, i)]\!]\eta\delta = [\![\mathsf{PVI}(\varphi, i)]\!]\eta\delta'$, then $\exists d \in \mathsf{sig}(\varphi) : \delta(d) \neq \delta'(d)$.*

Proof. We proceed by induction on φ.

- $\varphi = b$. Trivial.
- $\varphi = Y(e)$. In this case the two preconditions contradict, and the result trivially follows.
- $\varphi = \forall e\colon D.\psi$. Assume that $[\![\forall e\colon D.\psi]\!]\eta\delta \neq [\![\forall e\colon D.\psi]\!]\eta\delta'$, and furthermore, $\forall i \leqslant \mathsf{npred}(\forall e\colon D.\psi) : [\![\mathsf{PVI}(\forall e\colon D.\psi, i)]\!]\eta\delta = [\![\mathsf{PVI}(\forall e\colon D.\psi, i)]\!]\eta\delta'$.
  According to the semantics, we have $\forall u \in \mathbb{D}.[\![\psi]\!]\eta\delta[u/e] \neq \forall u' \in \mathbb{D}.[\![\psi]\!]\eta\delta'[u'/e]$, so $\exists u \in \mathbb{D}$ such that $[\![\psi]\!]\eta\delta_1[u/e] \neq [\![\psi]\!]\eta\delta_2[u/e]$. Choose an arbitrary such u. Observe that also for all $i \leqslant \mathsf{npred}(\psi)$, we know that $[\![\mathsf{PVI}(\psi, i)]\!]\eta\delta[u/e] = [\![\mathsf{PVI}(\psi, i)]\!]\eta\delta'[u/e]$. According to the induction hypothesis, $\exists d \in \mathsf{sig}(\psi)$ such that $\delta[u/e](d) \neq \delta'[u/e](d)$. Choose such a d, and observe that $d \neq e$ since otherwise $u \neq u$, hence $d \in \mathsf{sig}(\forall e\colon D.\psi)$, which is the desired result.
- $\varphi = \exists e\colon D.\psi$. Analogous to the previous case.
- $\varphi = \varphi_1 \wedge \varphi_2$. Assume that $[\![\varphi_1 \wedge \varphi_2]\!]\eta\delta \neq [\![\varphi_1 \wedge \varphi_2]\!]\eta\delta'$, and suppose that that for all $i \leqslant \mathsf{npred}(\varphi_1 \wedge \varphi_2)$, we know that $[\![\mathsf{PVI}(\varphi_1 \wedge \varphi_2, i)]\!]\eta\delta = [\![\mathsf{PVI}(\varphi_1 \wedge \varphi_2, i)]\!]\eta\delta'$. According to the first assumption, either $[\![\varphi_1]\!]\eta\delta \neq [\![\varphi_1]\!]\eta\delta'$, or $[\![\varphi_2]\!]\eta\delta \neq [\![\varphi_2]\!]\eta\delta'$. Without loss of generality, assume that $[\![\varphi_1]\!]\eta\delta \neq [\![\varphi_1]\!]\eta\delta'$, the other case is completely analogous. Observe that from our second assumption it follows that $\forall i \leqslant \mathsf{npred}(\varphi_1) : [\![\mathsf{PVI}(\varphi_1, i)]\!]\eta\delta = [\![\mathsf{PVI}(\varphi_1, i)]\!]\eta\delta'$. According to the induction hypothesis, we now find some $d \in \mathsf{sig}(\varphi_1)$ such that $\delta(d) \neq \delta'(d)$. Since $\mathsf{sig}(\varphi_1) \subseteq \mathsf{sig}(\varphi_1 \wedge \varphi_2)$, our result follows.
- $\varphi = \varphi_1 \vee \varphi_2$. Analogous to the previous case. □

This is now used in proving the following proposition, that shows that related signatures have the same solution. This result follows from the fact that R^M is a consistent correlation.

Proposition 1. *Let $\mathcal{E}$ be a PBES, with global control flow graph $(V, \rightarrow)$, and marking M. For all predicate environments η and data environments δ,*

$$(X, [\![\boldsymbol{v}]\!], [\![\boldsymbol{w}]\!]) R^M (Y, [\![\boldsymbol{v}']\!], [\![\boldsymbol{w}']\!]) \implies [\![\mathcal{E}]\!]\eta\delta(X(\boldsymbol{v}, \boldsymbol{w})) = [\![\mathcal{E}]\!]\eta\delta(Y(\boldsymbol{v}', \boldsymbol{w}')).$$

Proof. We show that R^M is a consistent correlation. The result then follows immediately from Theorem 1.

Let n be the smallest number such that for all $X, \boldsymbol{v}$, $M^n(X,\boldsymbol{v}) = M^n(X,\boldsymbol{v})$, and hence $M^n(X,\boldsymbol{v}) = M(X,\boldsymbol{v})$. Towards a contradiction, suppose that R^M is not a consistent correlation. Since R^M is not a consistent correlation, there exist $X, X', \boldsymbol{v}, \boldsymbol{v}', \boldsymbol{w}, \boldsymbol{w}'$ such that $(X, [\![\boldsymbol{v}]\!], [\![\boldsymbol{w}]\!]) R^{M^n} (X', [\![\boldsymbol{v}']\!], [\![\boldsymbol{w}']\!])$, and

$$\exists \eta \in \Theta_{R^{M^n}}, \delta : [\![\varphi_X]\!]\eta\delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}, [\![\boldsymbol{w}]\!]/\boldsymbol{d}] \neq [\![\varphi'_X]\!]\eta\delta[[\![\boldsymbol{v}']\!]/\boldsymbol{c}, [\![\boldsymbol{w}']\!]/\boldsymbol{d}].$$

By definition of R^{M^n}, $X = X'$, and $[\![\boldsymbol{v}]\!] = [\![\boldsymbol{v}']\!]$, hence this is equivalent to

$$\exists \eta \in \Theta_{R^{M^n}}, \delta : [\![\varphi_X]\!]\eta\delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}, [\![\boldsymbol{w}]\!]/\boldsymbol{d}] \neq [\![\varphi_X]\!]\eta\delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}, [\![\boldsymbol{w}']\!]/\boldsymbol{d}]. \tag{3}$$

Let η and δ be such, and let $\delta_1 = \delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}, [\![\boldsymbol{w}]\!]/\boldsymbol{d}]$ and $\delta_2 = \delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}, [\![\boldsymbol{w}']\!]/\boldsymbol{d}]$. Define $\varphi'_X = \mathsf{simplify}(\varphi_X[\boldsymbol{c} := \boldsymbol{v}])$. Since the values in $\boldsymbol{v}$ are closed, and from the definition of $\mathsf{simplify}$, we find that $[\![\varphi_X]\!]\eta\delta_1 = [\![\varphi'_X]\!]\eta\delta_1$, and likewise for δ_2. Therefore, we know that

$$[\![\varphi'_X]\!]\eta\delta_1 \neq [\![\varphi'_X]\!]\eta\delta_2. \tag{4}$$

Observe that for all $\boldsymbol{d}[k] \in M$, $[\![\boldsymbol{w}[k]]\!] = [\![\boldsymbol{w}'[k]]\!]$ by definition of R^M. Every predicate variable instance that might change the solution of φ'_X is a neighbour of $(X, \boldsymbol{v})$ in the control flow graph, according to Lemma 2. Take an arbitrary predicate variable instance $\mathsf{PVI}(\varphi_X, i) = Y(\boldsymbol{e}, \boldsymbol{e}')$ in φ'_X. We first show that $[\![\boldsymbol{e}'[\ell]]\!]\delta_1 = [\![\boldsymbol{e}'[\ell]]\!]\delta_2$ for all ℓ.

Observe that $[\![\boldsymbol{e}]\!]\delta_1 = [\![\boldsymbol{e}]\!]\delta_2$ since $\boldsymbol{e}$ are expressions substituted for control flow parameters, and hence are either constants, or the result of copying.

Furthermore, there is no unmarked parameter $\boldsymbol{d}[k]$ that can influence a marked parameter $\boldsymbol{d}[\ell]$ at location $(Y, \boldsymbol{u})$. If there is a $\boldsymbol{d}[\ell] \in M^n(Y, \boldsymbol{u})$ such that $\boldsymbol{d}[k] \in FV(\boldsymbol{e}'[\ell])$, and $\boldsymbol{d}[k] \notin M^n(X, \boldsymbol{v})$, then by definition of marking $\boldsymbol{d}[k] \in M^{n+1}(X, \boldsymbol{v})$, which contradicts the assumption that the marking is stable, so it follows that

$$[\![\boldsymbol{e}'[\ell]]\!]\delta_1 = [\![\boldsymbol{e}'[\ell]]\!]\delta_2 \text{ for all } \ell. \tag{5}$$

From (5), and since we have chosen the predicate variable instance arbitrarily, it follows that for all $1 \leqslant i \leqslant \mathsf{npred}(\varphi'_X)$, $[\![X(\boldsymbol{e}, \boldsymbol{e}')]\!]\eta\delta_1 = [\![X(\boldsymbol{e}, \boldsymbol{e}')]\!]\eta\delta_2$. Together with (4), according to Lemma 4, this implies that there is some $d \in \mathsf{sig}(\varphi'_X)$ such that $\delta_1(d) \neq \delta_2(d)$. From the definition of M^0, however, it follows that d must be marked in M^0, and hence also in M^n. According to the definition of R^{M^n} it then is the case that $\delta_1(d) = \delta_2(d)$, which is a contradiction. Since also in this case we derive a contradiction, the original assumption that R^M is not a consistent correlation does not hold, and we conclude that R^M is a consistent correlation. $\square$

A data parameter d that is irrelevant at location $(X, \boldsymbol{v})$ can be assigned a fixed default value $\mathsf{init}(d)$ in any predicate variable instance $\mathsf{PVI}(\varphi_Y, i)$ in any equation $Y \in \mathsf{bnd}(\mathcal{E})$ for which $\mathsf{pred}(\varphi_Y, i) = X$ and for which the control flow parameters have value $\boldsymbol{v}$. This is exactly what the function Reset, defined below, achieves.

Definition 19. *Let $\mathcal{E}$ be a PBES, let $(V, \rightarrow)$ be its control flow graph, with marking M. Resetting a PBES is inductively defined on the structure of $\mathcal{E}$.*

$$\begin{aligned}
\mathsf{Reset}_M(\epsilon) &= \epsilon \\
\mathsf{Reset}_M(\sigma X(\boldsymbol{c}\colon \boldsymbol{C}, \boldsymbol{d}\colon \boldsymbol{D}) = \varphi)\mathcal{E}') &= (\sigma \bar{X}(\boldsymbol{c}\colon \boldsymbol{C}, \boldsymbol{d}\colon \boldsymbol{D}) = \mathsf{Reset}_M(\varphi))\mathsf{Reset}_M(\mathcal{E}')
\end{aligned}$$

Resetting for formulae is defined inductively as follows:

$$\begin{aligned}
\mathsf{Reset}_M(b) &= b\\
\mathsf{Reset}_M(\varphi \wedge \psi) &= \mathsf{Reset}_M(\varphi) \wedge \mathsf{Reset}_M(\psi)\\
\mathsf{Reset}_M(\varphi \vee \psi) &= \mathsf{Reset}_M(\varphi) \vee \mathsf{Reset}_M(\psi)\\
\mathsf{Reset}_M(\forall d\colon D.\varphi) &= \forall d\colon D.\mathsf{Reset}_M(\varphi)\\
\mathsf{Reset}_M(\exists d\colon D.\varphi) &= \exists d\colon D.\mathsf{Reset}_M(\varphi)\\
\mathsf{Reset}_M(X(\boldsymbol{e}, \boldsymbol{e}')) &= \textstyle\bigwedge_{\boldsymbol{v} \in \mathsf{values}(\boldsymbol{c})}(\boldsymbol{e} = \boldsymbol{v} \implies \bar{X}(\boldsymbol{v}, \mathsf{Reset}^M_{(X,\boldsymbol{v})}(\boldsymbol{e}')))
\end{aligned}$$

With $\boldsymbol{e} = \boldsymbol{v}$ *we denote that for all* i, $\boldsymbol{e}[i] = \boldsymbol{v}[i]$. *The function* $\mathsf{Reset}^{(X,\boldsymbol{v})}_M(\boldsymbol{e}')$ *is defined positionally as follows:*

$$\mathsf{Reset}^{(X,\boldsymbol{v})}_M(\boldsymbol{e}')[i] = \begin{cases} \boldsymbol{e}'[i] & \textit{if } \boldsymbol{d}[i] \in M(X, \boldsymbol{v})\\ \mathsf{init}(\boldsymbol{d}[i]) & \textit{otherwise}. \end{cases}$$

Remark 1. We can reduce the number of equivalences we introduce in resetting a recurrence. This effectively reduces the guard as follows.

Let $X \in \mathsf{bnd}(\mathcal{E})$, such that $Y(\boldsymbol{e}, \boldsymbol{e}') = \mathsf{PVI}(\varphi_X, i)$, and let $I = \{j \mid \mathsf{dest}(X, i, j) = \bot\}$ denote the indices of the control flow parameters for which the destination is undefined.

Define $\boldsymbol{c}' = \boldsymbol{c}[i_1], \ldots \boldsymbol{c}[i_n]$ for $i_n \in I$, and $\boldsymbol{f} = \boldsymbol{e}[i_1], \ldots \boldsymbol{e}[i_n]$ to be the vectors of control flow parameters for which the destination is undefined, and the values that are assigned to them in predicate variable instance i. Observe that these are the only control flow parameters that we need to constrain in the guard while resetting.

We can redefine $\mathsf{Reset}_M(X(\boldsymbol{e}, \boldsymbol{e}'))$ as follows.

$$\mathsf{Reset}_{M_{loc}}(X(\boldsymbol{e}, \boldsymbol{e}')) = \bigwedge_{\boldsymbol{v}' \in \mathsf{values}(\boldsymbol{c}')} (\boldsymbol{f} = \boldsymbol{v}' \implies \bar{X}(\boldsymbol{v}, \mathsf{Reset}^{M_{loc}}_{(X,\boldsymbol{v})}(\boldsymbol{e}'))).$$

In this definition $\boldsymbol{v}$ is defined positionally as

$$\boldsymbol{v}[j] = \begin{cases} \boldsymbol{v}'[j] & \text{if } j \in I\\ \mathsf{dest}(X, i, j) & \text{otherwise} \end{cases}$$

Resetting irrelevant parameters preserves the solution of the PBES. We formalise this in Theorem 2 below. Our proof is based on consistent correlations. We first define the relation R^{Reset}, and we show that this is indeed a consistent correlation. Soundness then follows from Theorem 1. Note that R^{Reset} uses the relation R^M from Definition 18 to relate predicate variable instances of the original equation system. The latter is used in the proof of Lemma 6.

Definition 20. *Let* R^{Reset} *be the relation defined as follows.*

$$\begin{cases} X(\llbracket \boldsymbol{v} \rrbracket, \llbracket \boldsymbol{w} \rrbracket)) R^{\mathsf{Reset}} \bar{X}(\llbracket \boldsymbol{v} \rrbracket, \llbracket \mathsf{Reset}^{(X,\boldsymbol{v})}_M(\boldsymbol{w}) \rrbracket) & \\ X(\llbracket \boldsymbol{v} \rrbracket, \llbracket \boldsymbol{w} \rrbracket) R^{\mathsf{Reset}} X(\llbracket \boldsymbol{v} \rrbracket, \llbracket \boldsymbol{w}' \rrbracket) & \textit{if } X(\llbracket \boldsymbol{v} \rrbracket, \llbracket \boldsymbol{w} \rrbracket) R^M X(\llbracket \boldsymbol{v} \rrbracket, \llbracket \boldsymbol{w}' \rrbracket) \end{cases}$$

We first show that we can unfold the values of the control flow parameters in every predicate variable instance, by duplicating the predicate variable instance, and substituting the values of the CFPs.

Lemma 5. *Let η and δ be environments, and let $X \in \mathsf{bnd}(\mathcal{E})$, then for all $i \leqslant \mathsf{npred}(\varphi_X)$, such that $\mathsf{PVI}(\varphi_X, i) = Y(\boldsymbol{e}, \boldsymbol{e}')$,*

$$[\![Y(\boldsymbol{e}, \boldsymbol{e}'))]\!]\eta\delta = [\![\bigwedge_{\boldsymbol{v} \in \mathsf{values}(\boldsymbol{c})} (\boldsymbol{e} = \boldsymbol{v} \implies Y(\boldsymbol{v}, \boldsymbol{e}')]\!]\eta\delta$$

Proof. Straightforward; observe that $\boldsymbol{e} = \boldsymbol{v}$ for exactly one $\boldsymbol{v} \in \mathsf{values}(\boldsymbol{c})$, using that $\boldsymbol{v}$ is closed. $\square$

Next we establish that resetting irrelevant parameters is sound, *i.e.* it preserves the solution of the PBES. We first show that resetting a predicate variable instance in an R^{Reset}-correlating environment and a given data environment is sound.

Lemma 6. *Let $\mathcal{E}$ be a PBES, let $(V, \rightarrow)$ be its CFG, with marking M such that R^M is a consistent correlation, then*

$$\forall \eta \in \Theta_{R^{\mathsf{Reset}}}, \delta : [\![Y(\boldsymbol{e}, \boldsymbol{e}')]\!]\eta\delta = [\![\mathsf{Reset}_M(Y(\boldsymbol{e}, \boldsymbol{e}'))]\!]\eta\delta$$

Proof. Let $\eta \in \Theta_{R^{\mathsf{Reset}}}$, and δ be arbitrary. We derive this as follows.

$$\begin{array}{ll}
 & [\![\mathsf{Reset}_M(Y(\boldsymbol{e}, \boldsymbol{e}')))]\!]\eta\delta \\
= & \{\text{Definition 19}\} \\
 & [\![\bigwedge_{\boldsymbol{v} \in \mathsf{CFL}(Y)} (\boldsymbol{e} = \boldsymbol{v} \implies \bar{Y}(\boldsymbol{v}, \mathsf{Reset}_M^{(Y,\boldsymbol{v})}(\boldsymbol{e}')))]\!]\eta\delta \\
=^{\dagger} & \bigwedge_{\boldsymbol{v} \in \mathsf{CFL}(Y)} ([\![\boldsymbol{e}]\!]\delta = [\![\boldsymbol{v}]\!] \implies [\![\bar{Y}(\boldsymbol{v}, \mathsf{Reset}_M^{(Y,\boldsymbol{v})}(\boldsymbol{e}')]\!]\eta\delta)) \\
=^{\dagger} & \bigwedge_{\boldsymbol{v} \in \mathsf{CFL}(Y)} ([\![\boldsymbol{e}]\!]\delta = [\![\boldsymbol{v}]\!] \implies \eta(\bar{Y})([\![\boldsymbol{v}]\!]\delta, [\![\mathsf{Reset}_M^{(Y,\boldsymbol{v})}(\boldsymbol{e}')]\!]\delta)) \\
= & \{\eta \in \Theta_{R^{\mathsf{Reset}}}\} \\
 & \bigwedge_{\boldsymbol{v} \in \mathsf{CFL}(Y)} ([\![\boldsymbol{e}]\!]\delta = [\![\boldsymbol{v}]\!] \implies \eta(Y)([\![\boldsymbol{v}]\!]\delta, [\![\boldsymbol{e}']\!]\delta))) \\
=^{\dagger} & \bigwedge_{\boldsymbol{v} \in \mathsf{CFL}(Y)} ([\![\boldsymbol{e}]\!]\delta = [\![\boldsymbol{v}]\!] \implies [\![Y(\boldsymbol{v}, \boldsymbol{e}')]\!]\eta\delta)) \\
=^{\dagger} & [\![\bigwedge_{\boldsymbol{v} \in \mathsf{CFL}(Y)} (\boldsymbol{e} = \boldsymbol{v} \implies Y(\boldsymbol{v}, \boldsymbol{e}'))]\!]\eta\delta \\
= & \{\text{Lemma 5}\} \\
 & [\![Y(\boldsymbol{e}, \boldsymbol{e}'))]\!]\eta\delta
\end{array}$$

Here at † we have used the semantics. $\square$

By extending this result to the right-hand sides of equations, we can prove that R^{Reset} is a consistent correlation.

Proposition 2. *Let $\mathcal{E}$ be a PBES, and let $(V, \rightarrow)$ be a CFG, with marking M such that R^M is a consistent correlation. Let $X \in \mathsf{bnd}(\mathcal{E})$, with $\boldsymbol{v} \in CFL(X)$, then for all $\boldsymbol{w}$*

$$\forall \eta \in \theta_{R^{\mathsf{Reset}}}, \delta : [\![\varphi_X]\!]\eta\delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}, [\![\boldsymbol{w}]\!]/\boldsymbol{d}] = [\![\mathsf{Reset}_M(\varphi_X)]\!]\eta\delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}, [\![\mathsf{Reset}_M^{(X,\boldsymbol{v})}(\boldsymbol{w})]\!]/\boldsymbol{d}]$$

Proof. Let η and δ be arbitrary, and define $\delta_r = \delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}, [\![\mathsf{Reset}_M^{(X,\boldsymbol{v})}(\boldsymbol{w})]\!]/\boldsymbol{d}]$. We first prove that

$$[\![\varphi_X]\!]\eta\delta_r = [\![\mathsf{Reset}_M(\varphi_X)]\!]\eta\delta_r \qquad (6)$$

We proceed by induction on φ_X.

– $\varphi_X = b$. Since $\mathsf{Reset}_M(b) = b$ this follows immediately.
– $\varphi_X = Y(\boldsymbol{e})$. This follows immediately from Lemma 6.
– $\varphi_X = \forall y\colon D.\varphi$. We derive that $[\![\forall y\colon D.\varphi]\!]\eta\delta_r = \forall v \in \mathbb{D}.[\![\varphi]\!]\eta\delta_r[v/y]$. According to the induction hypothesis, and since we applied only a dummy transformation on y, we find that $[\![\varphi]\!]\eta\delta_r[v/y] = [\![\mathsf{Reset}_M(\varphi)]\!]\eta\delta_r[v/y]$, hence $[\![\forall y\colon D.\varphi]\!]\eta\delta_r = [\![\mathsf{Reset}_M(\forall y\colon D.\varphi)]\!]\eta\delta_r$.
– $\varphi_X = \exists y\colon D.\varphi$. Analogous to the previous case.
– $\varphi_X = \varphi_1 \wedge \varphi_2$. We derive that $[\![\varphi_1 \wedge \varphi_2]\!]\eta\delta_r = [\![\varphi_1]\!]\eta\delta_r \wedge [\![\varphi_2]\!]\eta\delta_r$. If we apply the induction hypothesis on both sides we get $[\![\varphi_1 \wedge \varphi_2]\!]\eta\delta_r = [\![\mathsf{Reset}_M(\varphi_1)]\!]\eta\delta_r \wedge [\![\mathsf{Reset}_M(\varphi_2)]\!]\eta\delta_r$. Applying the semantics, and the definition of Reset we find this is equal to $[\![\mathsf{Reset}_M(\varphi_1 \wedge \varphi_2)]\!]\eta\delta_r$.
– $\varphi_X = \varphi_1 \vee \varphi_2$. Analogous to the previous case.

Hence we find that $[\![\mathsf{Reset}_M(\varphi_X)]\!]\eta\delta_r = [\![\varphi_X]\!]\eta\delta_r$. It now follows immediately from the observation that R^M is a consistent correlation, and Definition 19, that $[\![\varphi_X]\!]\eta\delta_r = [\![\varphi_X]\!]\eta\delta[[\![\boldsymbol{v}]\!]/\boldsymbol{c}, [\![\boldsymbol{w}]\!]/\boldsymbol{d}]$. Our result follows by transitivity of $=$. $\square$

The theory of consistent correlations now gives an immediate proof of soundness of resetting irrelevant parameters, which is formalised by the following theorem.

Theorem 2. *Let $\mathcal{E}$ be a PBES, with control flow graph $(V, \to)$ and marking M. For all X, $\boldsymbol{v}$ and $\boldsymbol{w}$:*

$$[\![\mathcal{E}]\!](X([\![\boldsymbol{v}]\!], [\![\boldsymbol{w}]\!])) = [\![\mathsf{Reset}_M(\mathcal{E})]\!](\bar{X}([\![\boldsymbol{v}]\!], [\![\boldsymbol{w}]\!])).$$

Proof. Relation R^{Reset} is a consistent correlation, as witnessed by Proposition 2. From Theorem 1 the result now follows immediately. $\square$

The effect of resetting is that equations $\sigma X(\boldsymbol{d}\colon \boldsymbol{D}) = \varphi_X$ with different instances $\boldsymbol{v}$ for their formal parameters $\boldsymbol{d}$ may become more 'alike' after resetting, resulting in a potential reduction of the underlying Boolean equation system. This is nicely illustrated by applying the reset function on our running example.

Example 6. Applying Reset using the marking from Example 5 on the PBES of Example 1 results in the PBES below. Note that we have simplified the PBES slightly by removing the redundant conditions introduced by Reset, and by removing quantifiers that quantified over unused variables.

$$\begin{aligned}
&\nu\bar{X}(s\colon N, d\colon D) = (\forall e\colon D. s = 1 \implies \bar{Y}(2, e, e)) \wedge (s = 1 \implies \bar{X}(2, d_1))\\
&\quad \wedge (s = 2 \implies \bar{X}(3, d_1)) \wedge (s = 2 \implies \bar{X}(4, d_1)) \wedge (s = 3 \implies \bar{X}(1, d_1))\\
&\quad \wedge (s = 4 \implies \bar{X}(1, d_1))\\
&\mu\bar{Y}(s\colon N, d, v\colon D) = (s = 4 \implies d = v) \wedge (s = 2 \implies \bar{Y}(3, d_1, d_1))\\
&\quad \wedge (s = 2 \implies \bar{Y}(4, d, v)) \wedge (s = 3 \implies \bar{Y}(1, d_1, d_1)) \wedge (s = 4 \implies \bar{Y}(1, d_1, d_1))\\
&\mathbf{init}\ \bar{X}(1, d_1);
\end{aligned}$$

Note that $Y(2, d, v)$ depended on $Y(3, d, v)$; on the other hand, $\bar{Y}(2, d, v)$ depends on $\bar{Y}(3, d_1, d_1)$. In a way, the equations for $\bar{Y}(2, d, v)$ are more alike than those for $Y(2, d, v)$. This resemblance is also reflected in the size reduction after instantiation: for $|D| = 8$, the BES underlying the original PBES has 71 equations; the BES underlying the above PBES has 22 equations only.

5 Optimisation

The size of a CFG can grow exponentially in the number of control flow parameters in the worst case, hampering the effectiveness of our analysis. For instance, encoding the deadlock freedom model checking problem on a system consisting of N independent lossy buffers (from Example 1) results in a control flow graph with 4^N locations. To counter this, we study the local control flow and tailor our analysis to this new situation.

Drawing inspiration from [11], we next define a variant of the control flow graphs of Section 3 that do not suffer from this blow-up. This permits trading of the power of the subsequent data flow analysis and the computational complexity. In what follows, whenever applicable, we use terminology borrowed from [11].

From hereon, assume that $\mathcal{E}$ is a fixed PBES, $(\mathsf{source}, \mathsf{dest}, \mathsf{copy})$ is a unicity constraint, and the vector $\boldsymbol{c}$ is a vector of control flow parameters. The local control flow graph, which we define next, is a collection of unconnected graphs. Each connected subgraph represents a control flow parameter, and, in particular, its potential valuations, the equation in which it is considered, and which predicate variable instances it governs.

Definition 21. *The* local *control flow graph is a graph* $(V^{loc}, \hookrightarrow)$ *with:*

- $V^{loc} = \{(X, n, v) \mid X \in \mathsf{bnd}(\mathcal{E}) \wedge n \leqslant |\boldsymbol{c}| \wedge v \in \mathsf{values}(\boldsymbol{c}[n])\}$*, and*
- $\hookrightarrow \subseteq V^{loc} \times \mathbb{N} \times V^{loc}$ *is the least relation satisfying* $(X, n, v) \xhookrightarrow{i} (\mathsf{pred}(\varphi_X, i), n, w)$ *if:*
  1. $\mathsf{source}(X, i, n) = v$ *and* $\mathsf{dest}(X, i, n) = w$*, or*
  2. $\mathsf{source}(X, i, n) = \perp$*,* $\mathsf{pred}(\varphi_X, i) \neq X$ *and* $\mathsf{dest}(X, i, n) = w$*, or*
  3. $\mathsf{source}(X, i, n) = \perp$*,* $\mathsf{pred}(\varphi_X, i) \neq X$ *and* $\mathsf{copy}(X, i, n) = n$ *and* $v = w$*.*

Note that the size of a local control flow graph is linear in the number of equations, the number of control flow parameters and the size of the set of values each individual control flow parameter can assume.

The unicity constraint underlying a control flow parameter can be used to determine which predicate variable instances are potentially enabled in a location of the local control flow graph for that control flow parameter. In this case, we say that the predicate variable instance is *ruled* by this control flow parameter. From hereon, let $X \in \mathsf{bnd}(E)$ be an arbitrary bound predicate variable in the PBES $\mathcal{E}$ unless stated otherwise.

Definition 22. *Control flow parameter* $\boldsymbol{c}[j]$ rules $\mathsf{PVI}(\varphi_X, i)$ *whenever there is a value* v *for which* $(X, j, v) \xhookrightarrow{i}$.

The predicate variable instances that are potentially enabled by a control flow parameter form a *cluster* of predicate variable instances. These clusters will be used to guide us in our data flow analysis. We first identify which data parameters are potentially used in the scope of a predicate variable instance.

Definition 23. *Variable* d *is* used *for* $\mathsf{PVI}(\varphi_X, i)$ *if* $d \in FV(\mathsf{guard}^i(\varphi_X))$.

Parameters whose own values are potentially modified through a recursion are identified as *changed*. Observe that this only makes sense for self-recursions.

Definition 24. *Parameter* $\boldsymbol{d}[j] \in \mathsf{par}(X)$ *is* changed *for* $\mathsf{pred}(\varphi_X, i)$ *if both* $X = \mathsf{pred}(\varphi_X, i)$ *and* $\boldsymbol{d}[j] \neq \mathsf{data}(\varphi_X, i)[j]$.

We now formalise when a data parameter's data flow is entirely subsumed by a cluster. If this is the case, we say that such a data parameter *belongs to* the cluster.

Definition 25. *Let* c *be a control flow parameter and let* $d \in \mathsf{par}(X) \cap \mathcal{D}^{DP}$ *be a data parameter. We say that* d belongs to c *if either:*

1. d *is neither used nor changed for all* $\mathsf{PVI}(\varphi_X, i)$ *ruled by* c*, or*
2. *both the following hold:*
   - *whenever* d *is used in* $\mathsf{PVI}(\varphi_X, i)$*,* c *rules* $\mathsf{PVI}(\varphi_X, i)$*, and*
   - *whenever* d *is changed for* $\mathsf{PVI}(\varphi_X, i)$*,* c *rules* $\mathsf{PVI}(\varphi_X, i)$*.*

The set of data parameters that belong to c *is denoted by* $\mathsf{belongs}(c)$*.*

For ease of reasoning, we continue to work under the following assumption.

Assumption 2 *Each right-hand side predicate formula in a PBES contains at least one predicate variable instance and each data parameter in an equation belongs to at least one CFP; CFPs belong to no parameter.*

Observe that this assumption imposes no restrictions: equations $\sigma X(\boldsymbol{d} \colon \boldsymbol{D}) = \varphi_X$ where φ_X contains no predicate variable instances, can be strengthened to $\varphi_X \wedge X(\boldsymbol{d})$ in case $\sigma = \nu$ and weakened to $\varphi_X \vee X(\boldsymbol{d})$ otherwise, without affecting the solution to X or any of the other equations in an equation system. By adding a dummy parameter b of sort B to every equation $\sigma X(\boldsymbol{d} \colon \boldsymbol{D}) = \varphi_X$, initialising it to $true$, strenghtening each φ_X to $\varphi_X \wedge b = true$, and never changing b in predicate variable instances, we effectively turn b into a control flow parameter to which each data parameter can belong.

We identify relevant parameters in the local control flow graph in a way that is similar to how it is done in Section 4. First, in a control flow location (X, n, v), those parameters that belong to control flow parameter $\boldsymbol{c}[n]$ are marked that may be significant in $\varphi_X[\boldsymbol{c}[n] := v]$. Then, additional data parameters are identified as being relevant by determining whether they can (indirectly) affect a parameter that was already determined to be significant. For the soundness of the analysis, care must be taken that this also works in case a data parameter d that belongs to one control flow parameter affects a data parameter d' that belongs to another: in case the latter is already marked relevant, this requires that d is marked relevant too.

Definition 26. *Let* $(V^{loc}, \hookrightarrow)$ *be a local control flow graph for PBES* $\mathcal{E}$*. We define marking* $M_{loc} \colon V^{loc} \to \mathbb{P}(\mathcal{D}^{DP})$ *inductively as follows:*

$$
\begin{array}{ll}
M^0_{loc}(X,n,v) & = \{d \in \mathsf{belongs}(\boldsymbol{c}[n]) \mid d \in \mathsf{sig}(\mathsf{simplify}(\varphi_X[\boldsymbol{c}[n] := v]))\} \\
M^{k+1}_{loc}(X,n,v) & = M^k_{loc}(X,n,v) \\
 & \quad \cup \{d \in \mathsf{belongs}(\boldsymbol{c}[n]) \mid \exists i, w \text{ such that } \exists \boldsymbol{d}^Y[\ell] \in M^k_{loc}(Y,n,w) \\
 & \qquad (X,n,v) \xhookrightarrow{i} (Y,n,w) \wedge d \text{ affects } \mathsf{data}(i)[\ell]\} \\
 & \quad \cup \{d \in \mathsf{belongs}(\boldsymbol{c}[n]) \mid \exists i, m, w \text{ such that } \exists \boldsymbol{d}^Y[\ell] \in M^k_{loc}(Y,m,w) \\
 & \qquad \boldsymbol{d}^Y[\ell] \notin \mathsf{belongs}(\boldsymbol{c}[n]) \wedge Y = \mathsf{pred}(\varphi_X, i) \wedge d \text{ affects } \mathsf{data}(i)[\ell]\} \\
M_{loc}(X,n,v) & = \bigcup_{k \in \mathbb{N}} M^k_{loc}(X,n,v)
\end{array}
$$

The local marking can again be used to reset data parameters using function Reset, combined with the induced marking $M_{loc}(X, \boldsymbol{v})$, defined as $d \in M_{loc}(X, \boldsymbol{v})$ iff for all k for which $d \in \mathsf{belongs}(\boldsymbol{c}[k])$ we have $d \in M_{loc}(X, k, \boldsymbol{v}[k])$. This induced marking overapproximates the marking computed in Section 4.

Lemma 7. *Let, for given PBES $\mathcal{E}$, $(V, \to)$ be a global control flow graph with marking M, and let $(V^{loc}, \hookrightarrow)$ be a local control flow graph with induced marking M_{loc}. Then $M(X, \boldsymbol{v}) \subseteq M_{loc}(X, \boldsymbol{v})$ for all $(X, \boldsymbol{v})$.*

Proof. We prove the more general statement that for all natural numbers n it holds that $\forall (X, \boldsymbol{v}) \in V, \forall d \in M^n(X, \boldsymbol{v}) : (\forall j : d \in \mathsf{belongs}(\boldsymbol{c}[j]) \implies d \in M^n_{loc}(X, j, \boldsymbol{v}[j]))$. The lemma then is an immediate consequence.

We proceed by induction on n.

- $n = 0$. Let $(X, \boldsymbol{v})$ and $d \in M^0(X, \boldsymbol{v})$ be arbitrary. We need to show that $\forall j : d \in \mathsf{belongs}(\boldsymbol{c}[j]) \implies d \in M^0_{loc}(X, j, \boldsymbol{v}[j])$.
  Let j be arbitrary such that $d \in \mathsf{belongs}(\boldsymbol{c}[j])$. Since $d \in M^0(X, \boldsymbol{v})$, by definition $d \in \mathsf{sig}(\mathsf{simplify}(\varphi_X[\boldsymbol{c} := \boldsymbol{v}]))$, hence also $d \in \mathsf{sig}(\mathsf{simplify}(\varphi_X[\boldsymbol{c}[j] := \boldsymbol{v}][j]))$. Combined with the assumption that $d \in \mathsf{belongs}(\boldsymbol{c}[j])$, this gives us $d \in M^0_{loc}(X, j, \boldsymbol{v}[j])$ according to Definition 21.
- $n = m + 1$. As induction hypothesis assume that $\forall (X, \boldsymbol{v}) \in V : \forall d : d \in M^m(X, \boldsymbol{v}) \implies (\forall j : d \in \mathsf{belongs}(\boldsymbol{c}[j]) \implies d \in M^m_{loc}(X, j, \boldsymbol{v}[j]))$. Now let $(X, \boldsymbol{v})$ be arbitrary, and let $d \in M^{m+1}(X, \boldsymbol{v})$. Also let j be arbitrary, and assume that $d \in \mathsf{belongs}(\boldsymbol{c}[j])$.
  We need to show that $d \in M^{m+1}_{loc}(X, j, \boldsymbol{v}[j])$. We proceed by distinction on the cases of Definition 17. If $d \in M^m(X, \boldsymbol{v})$ the result follows immediately from the induction hypothesis.
  Now suppose there is an $i \leqslant \mathsf{npred}(\varphi_X)$ such that $(X, \boldsymbol{v}) \xrightarrow{i} (\mathsf{pred}(\varphi_X, i), \boldsymbol{w})$, and there is some $\boldsymbol{d}[\ell] \in M^m(\mathsf{pred}(\varphi_X, i), \boldsymbol{w})$ with $d \in FV(\mathsf{data}(\varphi_X, i)[\ell])$
  Let i and $\boldsymbol{d}[\ell]$ be such.
  According to the induction hypothesis, $\forall k : \boldsymbol{d}[\ell] \in \mathsf{belongs}(\boldsymbol{c}[k]) \implies \boldsymbol{d}[\ell] \in M^m_{loc}(\mathsf{pred}(\varphi_X, i), k, \boldsymbol{w}[k])$.
  - $\boldsymbol{d}[\ell]$ belongs to $\boldsymbol{c}[j]$. According to the induction hypothesis we have $\boldsymbol{d}[\ell] \in M^m_{loc}(\mathsf{pred}(\varphi_X, i), j, \boldsymbol{w}[j])$. We have $d \in FV(\mathsf{data}(\varphi_X, i)[\ell])$, so we only need to show that $(X, j, \boldsymbol{v}[j]) \overset{i}{\hookrightarrow} (\mathsf{pred}(\varphi_X, i), j, \boldsymbol{w}[j])$. We distinguish the cases for j from Definition 14.
    * $\mathsf{source}(X, i, j) = \boldsymbol{v}[j]$ and $\mathsf{dest}(X, i, j) = \boldsymbol{w}[j]$, then the edge $(X, j, \boldsymbol{v}[j]) \overset{i}{\hookrightarrow} (\mathsf{pred}(\varphi_X, i), j, \boldsymbol{w}[k])$ also exists according to Definition 21.
    * $\mathsf{source}(X, i, j) = \perp$, $\mathsf{copy}(X, i, j) = j$ and $\boldsymbol{v}[j] = \boldsymbol{w}[j]$. In case $\mathsf{pred}(\varphi_X, i) \neq X$ the edge exists locally, and we are done. Now suppose that $\mathsf{pred}(\varphi_X, i) = X$. Then $\mathsf{PVI}(\varphi_X, i)$ is not ruled by $\boldsymbol{c}[j]$. Furthermore, $\boldsymbol{d}[\ell]$ is changed in $\mathsf{PVI}(\varphi_X, i)$, hence $\boldsymbol{d}[\ell]$ cannot belong to $\boldsymbol{c}[j]$, which is a contradiction.
    * $\mathsf{source}(X, i, j) = \perp$, $\mathsf{copy}(X, i, j) = \perp$ and $\mathsf{dest}(X, i, j) = \boldsymbol{w}[j]$. This is completely analogous to the previous case.

- $\boldsymbol{d}[\ell]$ does not belong to $\boldsymbol{c}[j]$. Recall that there must be some $\boldsymbol{c}[k]$ such that $\boldsymbol{d}[\ell]$ belongs to $\boldsymbol{c}[k]$, and by assumption now $\boldsymbol{d}[\ell]$ does not belong to $\boldsymbol{c}[j]$. Then according to Definition 26, d is marked in $M_{loc}^{m+1}(X, j, \boldsymbol{v}[j])$, which is what we need to prove.

The marking induced by the local analysis can again be used to reset data parameters without affecting the solution to the equation system.

Theorem 3. *Let $\mathcal{E}$ be a PBES, with local control flow graph $(V^{loc}, \hookrightarrow)$ and induced marking M_{loc}. Then for all X, $\boldsymbol{v}$ and $\boldsymbol{w}$: $[\![\mathcal{E}]\!](X(\boldsymbol{v}, \boldsymbol{w})) = [\![\mathsf{Reset}_{M_{loc}}(\mathcal{E})]\!](\bar{X}(\boldsymbol{v}, \boldsymbol{w}))$.*

The correctness of the above theorem follows from Lemma 7 and Theorem 2.

6 Case studies

We implemented our techniques in the tool `pbesstategraph` of the mCRL2 toolset [2]. Here, we report on the tool's effectiveness in simplifying the PBESs originating from model checking problems and behavioural equivalence checking problems: we compare sizes of the BESs underlying the original PBESs to those underlying the PBESs obtained after running the tool `pbesparelm` (which implements the techniques from [9]) to those underlying the PBESs obtained after running our tool.

Our cases are taken from the literature. For the model checking problems, we considered the *Onebit* protocol, which is a complex sliding window protocol, and Hesselink's handshake register [6]. Both protocols are parametric in the set of values that can be read and written. A selection of properties of varying complexity and varying nesting degree, expressed in the data-enhanced modal μ-calculus are checked.[2] For the behavioural equivalence checking problems, we considered a number of communication protocols such as the *Alternating Bit Protocol* (ABP), the *Concurrent Alternating Bit Protocol* (CABP), a two-place buffer (Buf) and the aforementioned Onebit protocol. Moreover, we compare an implementation of Hesselink's register to a specification of the protocol that is correct with respect to trace equivalence (but for which currently no PBES encoding exists) but not with respect to the two types of behavioural equivalence checking problems we consider here: branching bisimilarity and weak bisimilarity.

Our experiments confirm that our technique can achieve as much as an additional reduction of about 90% over `pbesparelm`, see the model checking problems and equivalence checking problems for Hesselink's register. Compared to the sizes of the BESs underlying the original PBESs, the reductions can be immense.

7 Conclusions and future work

We described a new static analysis technique for PBESs that employs a notion of control flow to determine when data parameters become irrelevant. Using this information, the PBES can be simplified, leading to smaller underlying Boolean equation systems.

[2] The formulae are contained in the appendix; here we simply use textual characterisations instead.

Table 1. Sizes of the BESs underlying the original PBESs, the PBESs reduced using `pbesparelm` and the PBESs reduced using `pbesstategraph`. The numbers reported reflect the number of equations generated using instantiation. Verdict $\surd$ means the outcome of the verification problem is $true$; $\times$ means the outcome is $false$.

		Original	`pbesparelm`	`pbesstategraph`	verdict
Model Checking Problems					
No deadlock					
Onebit	$\lvert D\rvert = 2$	81 921	11 409	9 089	$\surd$
	$\lvert D\rvert = 3$	289 297	11 409	9 089	$\surd$
	$\lvert D\rvert = 4$	742 401	11 409	9 089	$\surd$
Hesselink	$\lvert D\rvert = 2$	540 737	2 065	2 065	$\surd$
	$\lvert D\rvert = 3$	13 834 801	2 065	2 065	$\surd$
No spontaneous generation of messages					
Onebit	$\lvert D\rvert = 2$	185 089	30 593	22 145	$\surd$
	$\lvert D\rvert = 3$	1 278 433	57 553	39 169	$\surd$
	$\lvert D\rvert = 4$	5 588 481	92 289	60 161	$\surd$
Messages that are read are inevitably sent					
Onebit	$\lvert D\rvert = 2$	153 985	57 553	41 473	$\times$
	$\lvert D\rvert = 3$	579 745	115 489	78 817	$\times$
	$\lvert D\rvert = 4$	1 549 057	192 865	127 233	$\times$
Messages can overtake one another					
Onebit	$\lvert D\rvert = 2$	164 353	61 441	44 609	$\times$
	$\lvert D\rvert = 3$	638 065	127 153	88 225	$\times$
	$\lvert D\rvert = 4$	1 735 681	216 193	146 049	$\times$
Values written to the register can be read					
Hesselink	$\lvert D\rvert = 2$	1 093 761	1 081 345	89 089	$\surd$
	$\lvert D\rvert = 3$	27 876 961	27 656 641	561 169	$\surd$
Equivalence Checking Problems					
Branching bisimulation equivalence					
ABP-CABP	$\lvert D\rvert = 2$	31 265	31 265	30 225	$\surd$
	$\lvert D\rvert = 4$	73 665	73 665	69 681	$\surd$
Buf-Onebit	$\lvert D\rvert = 2$	844 033	706 561	511 554	$\surd$
	$\lvert D\rvert = 4$	8 754 689	5 939 201	3 707 138	$\surd$
Hesselink I-S	$\lvert D\rvert = 2$	21 062 529	21 062 529	1 499 714	$\times$
Weak bisimulation equivalence					
ABP-CABP	$\lvert D\rvert = 2$	50 713	49 617	47 481	$\surd$
	$\lvert D\rvert = 4$	117 337	113 361	106 089	$\surd$
Buf-Onebit	$\lvert D\rvert = 2$	966 897	706 033	552 226	$\surd$
	$\lvert D\rvert = 4$	9 868 225	5 869 505	3 862 402	$\surd$
Hesselink I-S	$\lvert D\rvert = 2$	29 868 273	28 579 137	2 067 650	$\times$

Compared to existing techniques, our new static analysis technique can lead to additional reductions of up-to 90% in extreme cases, as illustrated by our experiments.

Several techniques described in this paper can be used to enhance existing reduction techniques for PBESs. For instance, our notion of a *guard* of a predicate variable instance in a PBES can be put to use to cheaply improve on the heuristics for constant elimination [9]. Moreover, we believe that our (re)construction of control flow graphs from PBESs can be used to automatically generate invariants for PBESs. The theory on invariants for PBESs is well-established, but still lacks proper tool support.

References

1. A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, 1986.
2. S. Cranen, J. F. Groote, J. J. A. Keiren, F. P. M. Stappers, E. P. de Vink, W. Wesselink, and T. A. C. Willemse. An overview of the mCRL2 toolset and its recent advances. In *TACAS*, volume 7795 of *LNCS*, pages 199–213. Springer, 2013.
3. J.-C. Fernandez, M. Bozga, and L. Ghirvu. State space reduction based on live variables analysis. *Science of Computer Programming*, 47(2-3):203–220, 2003.
4. J. F. Groote and B. Lisser. Computer assisted manipulation of algebraic process specifications. *ACM SIGPLAN Notices*, 37(12):98, 2002.
5. N. Hentze and D. McAllester. Linear-time subtransitive control flow analysis. In *PLDI '97*, pages 261–272. ACM, 1997.
6. W. H. Hesselink. Invariants for the construction of a handshake register. *Information Processing Letters*, 68:173–177, 1998.
7. A. Mader. *Verification of Modal Properties Using Boolean Equation Systems*. PhD thesis, Technische Universität München, 1997.
8. R. Mateescu. *Vérification des propriétés temporelles des programmes parallèles*. PhD thesis, Institut National Polytechnique de Grenoble, 1998.
9. S. Orzan, J. W. Wesselink, and T. A. C. Willemse. Static Analysis Techniques for Parameterised Boolean Equation Systems. In *TACAS'09*, volume 5505 of *LNCS*, pages 230–245. Springer, 2009.
10. B. Ploeger, W. Wesselink, and T.A.C. Willemse. Verification of reactive systems via instantiation of parameterised Boolean equation systems. *Information and Computation*, 209(4):637–663, 2011.
11. J. C. van de Pol and M. Timmer. State Space Reduction of Linear Processes. In *ATVA*, volume 5799 of *LNCS*, pages 54–68. Springer, 2009.
12. T. A. C. Willemse. Consistent Correlations for Parameterised Boolean Equation Systems with Applications in Correctness Proofs for Manipulations. In *CONCUR 2010*, volume 6269 of *LNCS*, pages 584–598. Springer, 2010.
13. K. Yorav and O. Grumberg. Static Analysis for State-Space Reductions Preserving Temporal Logics. *Formal Methods in System Design*, 25(1):67–96, 2004.

A μ-calculus formulae

Below, we list the formulae that were verified in Section 6. All formulae are denoted in the the first order modal μ-calculus, an mCRL2-native data extension of the modal μ-calculus. The formulae assume that there is a data specification defining a non-empty sort D of messages, and a set of parameterised actions that are present in the protocols. The scripts we used to generate our results, and the complete data of the experiments are available from `https://github.com/jkeiren/pbesstategraph-experiments`

A.1 Onebit protocol verification

– No deadlock:

$$\nu X.[true]X \wedge \langle true\rangle true$$

Invariantly, over all reachable states at least one action is enabled.

– Messages that are read are inevitably sent:

$$\nu X.[true]X \wedge \forall d\colon D.[ra(d)]\mu Y.([\overline{sb(d)}]Y \wedge \langle true\rangle true))$$

The protocol receives messages via action ra and tries to send these to the other party. The other party can receive these via action sb.

– Messages can be overtaken by other messages:

$$\begin{array}{l}\mu X.\langle true\rangle X \vee \exists d\colon D.\langle ra(d)\rangle\mu Y.\\ \quad\Big(\langle\overline{sb(d)}\rangle Y \vee \exists d'\colon D.d \neq d' \wedge \langle ra(d')\rangle\mu Z.\\ \qquad(\langle\overline{sb(d)}\rangle Z \vee \langle sb(d')\rangle true)\\ \quad\Big)\end{array}$$

That is, there is a trace in which message d is read, and is still in the protocol when another message d' is read, which then is sent to the receiving party before message d.

– No spontaneous messages are generated:

$$\begin{array}{l}\nu X.[\overline{\exists d\colon D.ra(d)}]X \wedge\\ \quad\forall d'\colon D.[ra(d')]\nu Y(m_1\colon D = d').\\ \qquad\Big([\overline{\exists d\colon D.ra(d) \vee sb(d)}]Y(m_1) \wedge\\ \qquad\ \forall e\colon D.[sb(e)]((m_1 = e) \wedge X) \wedge\\ \qquad\ \forall e'\colon D.[ra(e')]\nu Z(m_2\colon D = e').\\ \qquad\quad\Big([\overline{\exists d\colon D.ra(d) \vee sb(d)}]Z(m_2) \wedge\\ \qquad\quad\ \forall f\colon D.[sb(f)]((f = m_1) \wedge Y(m_2))\\ \qquad\quad\Big)\\ \qquad\Big)\end{array}$$

Since the onebit protocol can contain two messages at a time, the formula states that only messages that are received can be subsequently sent again. This requires storing messages that are currently in the buffer using parameters m_1 and m_2.

A.2 Hesselink's register

- No deadlock:

$$\nu X.[true]X \wedge \langle true \rangle true$$

- Values that are written to the register can be read from the register if no other value is written to the register in the meantime.

$$
\begin{array}{l}
\nu X.[true]X \wedge \forall w\colon D.[begin\_write(w)]\nu Y. \\
\qquad\qquad \Big([\overline{end\_write}]Y \wedge [end\_write]\nu Z. \\
\qquad\qquad\qquad \Big([\overline{\exists d : D.begin\_write(d)}]Z \wedge [begin\_read]\nu W. \\
\qquad\qquad\qquad\qquad ([\overline{\exists d : D.begin\_write(d)}]W \wedge \\
\qquad\qquad\qquad\qquad\qquad \forall w' : D.[end\_read(w')](w = w')) \\
\qquad\qquad\qquad \Big) \\
\qquad\qquad \Big)
\end{array}
$$